# Towards Large Language Models at the Edge on Mobile, Augmented Reality, and Virtual Reality Devices with Unity

## ABSTRACT

This year, there has been a surge in applications powered by Large Language Models (LLMs). Modern LLMs often demand extensive memory and computation for inference. However, ongoing research in quantization and model distillation continues to reduce these processing requirements. Alongside advancements in hardware, it's now becoming feasible to run LLMs on mobile devices, Augmented Reality (AR), and Virtual Reality (VR) platforms. This paper explores developing state-of-the-art LLAMA2 LLM's models across various devices, including laptops, mobiles, AR, and VR systems. We introduce a Unity-based project that enables the compilation of the LLAMA2 model for diverse platforms, marking a novel contribution to the field. Furthermore, we benchmark performance, share best practices to optimize token generation rates and discuss potential future directions.

## 1. INTRODUCTION

The landscape of technology and artificial intelligence has been rapidly reshaped by the advent of Large Language Models (LLMs). Their capabilities, ranging from natural language processing to complex task completions, have ushered in a new era of applications and tools that stand at the forefront of innovation. Yet, the intrinsic complexity of these models, often requiring vast amounts of computational power and memory, has historically restricted their deployment to high-end servers and data centers.

However, a paradigm shift is on the horizon. As the demand for on-the-go intelligence grows, so does the need to bring the power of LLMs to edge devices. These include everyday mobile devices, as well as more specialized platforms like Augmented Reality (AR) and Virtual Reality (VR) systems. The benefits are manifold: real-time processing, reduced data transfer, enhanced user privacy, and the ability to operate without constant internet connectivity.

Recent advancements in quantization and model distillation techniques are making this dream a closer reality, progressively lowering the computational footprint of LLMs. Concurrently, the hardware industry is not lagging, with significant leaps in the processing capabilities of portable devices.

Within this context, our study dives into the potential of the state-of-the-art LLAMA2 LLM. We aim to assess its adaptability and performance across a range of platforms, especially focusing on how it fares in the ever-evolving mobile, AR, and VR ecosystems. By utilizing a Unity-based compilation technique, we hope to bridge the gap between complex LLM architectures and varied device specifications.

## 2. BACKGROUND AND RELATED WORK

The history of computational linguistic models traces back to the mid-20th century with ELIZA, one of the first ever computer programs designed for simulating typed conversation. Developed by Joseph Weizenbaum in 1966, ELIZA was a groundbreaking endeavor that emulated Rogerian therapist, primarily through pattern matching and substitution methodology [14]. Though rudimentary by today's standards, ELIZA set the stage for the exploration of man-machine communication via natural language.

Fast forward a few decades, and the landscape of natural language processing experienced a seismic shift with models like BERT, which utilized deep bidirectional transformers for enhanced language understanding [4]. Following BERT, models such as LLAMA2 were introduced, signifying marked advancements in capabilities [12]. As these models became increasingly sophisticated, they have found application across a diverse range of industries. However, deploying these models, especially on edge devices, presented challenges due to their computational demands.

Quantization naturally was viewed as one approach to address these challenges. Early neural network quantization work by Jacob et al. [9] was later followed by efforts that continue to focus on improving the efficiency of transformers and large language models [1,2,3]. Recently, GPTQ introduced a novel one-shot weight quantization method for GPT models, addressing their computational and storage challenges by compressing large models with minimal accuracy loss while maintaining the essence of their original capabilities [6].

In recent times, the focus has been on enhancing the portability and accessibility of foundational LLM research across various devices. Leading this initiative is the llama.cpp library, which facilitates the execution of Llama2 on a diverse range of devices [8]. Augmenting this, llama.cpp-dotnet brought C# wrappers into the mix, extending the reach of the Llama2 suite of LLMs into the .NET landscape [11]. llama.cpp leverages ggml [7], a C-oriented tensor library specifically designed for machine learning, aiming to maximize model performance on conventional hardware.

The Unity Engine [13] stands out as a versatile multiplatform development tool, bridging the gap between innovative ideas and diverse platforms. It empowers developers to craft immersive experiences that seamlessly transition from Android smartphones to the captivating realms of virtual reality

(VR) and augmented reality (AR). Such adaptability ensures that content reaches a wider audience, no matter their device or platform of choice, reaffirming Unity's pivotal role in modern application development.

## 3. UNITY.LLAMA

### 3.1 Integrating LLMs and Unity

Unity.LLAMA connects the capabilities of the LLAMA2 LLM with Unity's diverse platform support. The aim is to make advanced language models available across various devices, from mobile phones to AR and VR systems, using Unity's well-established cross-platform features.

### 3.2 Implementation Details

The core of Unity.LLAMA's integration is the dotnet-llamacpp library. This library, designed to link the LLM with .NET ecosystems, was modified to meet the .NET Standard 2.1 specification, facilitating its use within Unity.

### 3.3 Device-Specific Optimizations

Different devices have varied capabilities, which necessitated tailored approaches for each platform. Unity.LLAMA is designed to load model parts dynamically, considering the device's computational and memory resources.

On mobile devices, which often have more limited resources than desktop systems. For performance, Unity.LLAMA supports asynchronous operations and a streamlined rendering process to give users faster results.

### 3.4 User Interface & Experience

Unity.LLAMA isn't just about backend operations. The research interface we've provided allows users to explore and test the system in various ways. One example project integrates whisper.cpp, letting users issue voice commands to the LLAMA2 model. The interface is designed to be accessible, whether it's on a mobile device or a VR headset.

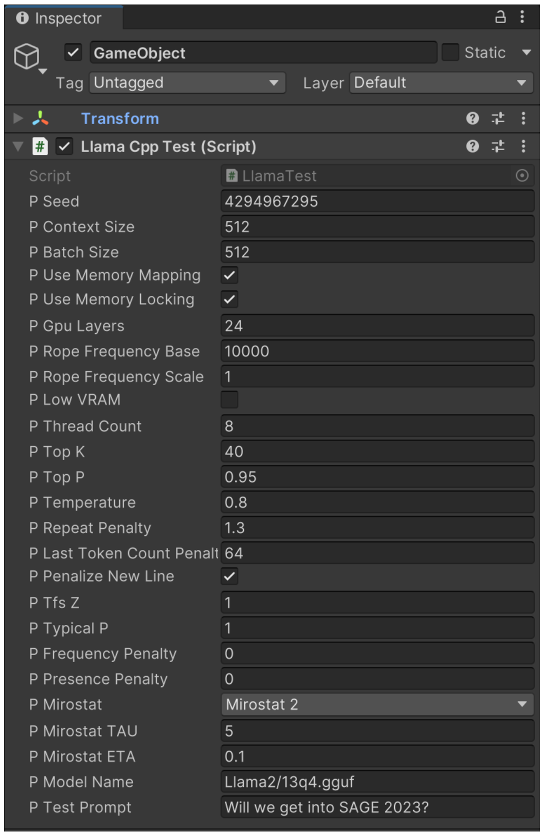
## 4. RESULTS

### 4.1 AR/VR Voice Assistant

As a reference project for bringing the capabilities of Large Language Models to practical applications, we developed a voice assistant application by leveraging both whisper.cpp for voice command processing and llama.cpp for natural language understanding. This integration allowed users to communicate with the Unity environment interactively and can be found at the path below.

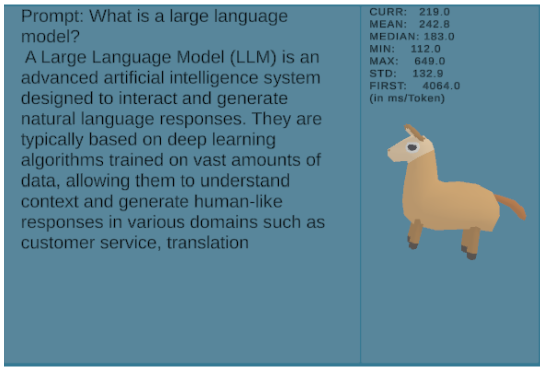
```
Assets\
    Scenes\
        SageScene.scene
        ...
```

When users pose questions or commands vocally, whisper.cpp captures and processes the audio data, converting spoken words into text. This transcribed text is then fed to the Llama2 model via llama.cpp to generate relevant responses.



| | |
|---|---|
| GameObject | Static |
| Tag | Untagged |
| Layer | Default |
| Script | LlamaTest |
| P Seed | 4294967295 |
| P Context Size | 512 |
| P Batch Size | 512 |
| P Use Memory Mapping | ✓ |
| P Use Memory Locking | ✓ |
| P Gpu Layers | 24 |
| P Rope Frequency Base | 10000 |
| P Rope Frequency Scale | 1 |
| P Low VRAM | |
| P Thread Count | 8 |
| P Top K | 40 |
| P Top P | 0.95 |
| P Temperature | 0.8 |
| P Repeat Penalty | 1.3 |
| P Last Token Count Penalt | 64 |
| P Penalize New Line | ✓ |
| P Tfs Z | 1 |
| P Typical P | 1 |
| P Frequency Penalty | 0 |
| P Presence Penalty | 0 |
| P Mirostat | Mirostat 2 |
| P Mirostat TAU | 5 |
| P Mirostat ETA | 0.1 |
| P Model Name | Llama2/13q4.gguf |
| P Test Prompt | Will we get into SAGE 2023? |

**Figure 1: Model parameters exposed to unity inspector**



Prompt: What is a large language model?
A Large Language Model (LLM) is an advanced artificial intelligence system designed to interact and generate natural language responses. They are typically based on deep learning algorithms trained on vast amounts of data, allowing them to understand context and generate human-like responses in various domains such as customer service, translation

```
CURR:   219.0
MEAN:   242.8
MEDIAN: 183.0
MIN:    112.0
MAX:    649.0
STD:    132.9
FIRST:  4064.0
(in ms/Token)
```

**Figure 2: Reference Project scene answering user query**

### 4.2 Model Selection

As the application does not have a simple performance metric, we ran models that gave qualitatively reasonable results and compared timing performance below on a Core(TM) i9-12900H, 32GB PC. The Models used can be found at huggingface's model repository [5, 10] and utilize llama.cpp for quantization. A key for the quantization methods used is provided below:

- Q2_K - 2-bit quantization
- Q3_K_M - 3-bit quantization
- Q4_K_M - 4-bit quantization

| Llama2 7B Benchmarks | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Model** | **RAM GB** | **Threads** | **Mean** | **Med** | **Min** | **Max** | **STD** |
| Q2_K | 5.33 | 2 | 294.9 | 294 | 268 | 322 | 8.9 |
| Q3_K_M | 5.8 | 2 | 294.5 | 293.5 | 267 | 468 | 12.5 |
| Q4_K_M | 6.58 | 2 | 302.7 | 302 | 284 | 351 | 11.2 |
| Q5_K_M | 7.28 | 2 | 351.7 | 351 | 327 | 395 | 10.8 |
| Q8_0 | 9.66 | 2 | 421.9 | 420 | 394 | 466 | 10.1 |
| Q2_K | 5.33 | 4 | 242.2 | 241 | 222 | 261 | 7.8 |
| Q3_K_M | 5.8 | 4 | 243.5 | 243 | 215 | 268 | 8.8 |
| Q4_K_M | 6.58 | 4 | 223 | 225 | 205 | 235 | 7.3 |
| Q5_K_M | 7.28 | 4 | 278.3 | 279 | 251 | 300 | 7.8 |
| Q8_0 | 9.66 | 4 | 314.5 | 315 | 301 | 324 | 5 |
| Q2_K | 5.33 | 8 | 169.8 | 169 | 160 | 185 | 4.3 |
| Q3_K_M | 5.8 | 8 | 172 | 171 | 162 | 190 | 4.1 |
| Q4_K_M | 6.58 | 8 | 160.1 | 160 | 142 | 178 | 5.5 |
| Q5_K_M | 7.28 | 8 | 196.8 | 196 | 185 | 222 | 4.9 |
| Q8_0 | 9.66 | 8 | 235 | 234 | 227 | 248 | 4.2 |
| Q2_K | 5.33 | 16 | 136.3 | 135 | 124 | 232 | 8.8 |
| Q3_K_M | 5.8 | 16 | 141 | 140 | 127 | 259 | 9 |
| Q4_K_M | 6.58 | 16 | 139.3 | 139 | 121 | 167 | 6.4 |
| Q5_K_M | 7.28 | 16 | 162.4 | 161 | 146 | 208 | 7.2 |
| Q8_0 | 9.66 | 16 | 200.2 | 199 | 183 | 243 | 7.6 |
| **Llama2 13B Benchmarks** | | | | | | | |
| Q2_K | 7.93 | 16 | 230 | 230 | 217 | 257 | 7.4 |
| Q3_K_M | 8.84 | 16 | 232 | 233 | 204 | 298 | 16.9 |
| Q4_K_M | 10.37 | 16 | 252.6 | 251 | 233 | 397 | 12.7 |
| Q5_K_M | 11.73 | 16 | 297.9 | 297 | 276 | 429 | 13.8 |
| Q8_0 | 16.33 | 16 | 402.1 | 390 | 361 | 692 | 39 |

**Table 1: Benchmarks Taken from Core(TM) i9-12900H, 32GB PC. Mean, Med, Min, Max, STD are all measured in milliseconds per token.**

- Q5_K_M - 5-bit quantization
- Q8_0 - 8-bit quantization

Upon analyzing the results from table 1, we identified the LLama7B_Q4_K_M model as the most suitable for deployment on the 12GB Snapdragon XR2+ processor found in the Meta Quest 2. The rational comes down to the Q2 models being unreliable in qualitative performance. Therefore, given that perplexity rises as we further quantize, it appears that 4-bit quantization consistently strikes the best balance between speed improvement and reduced memory usage. Although the 13B model may offer higher performance, there was no noticeable qualitative difference in the context of our simple Q&A application. Moreover, the 13B model demanded significantly more memory.

## 4.3 Parameters for Inference

Parameters used are defined below:

```
Context Size                512
Batch Size                  512
Memory Mapping              True
Memory Locking              True
Gpu Layers                  0
Rope Frequency Base         10000
Rope Frequency Scale        1
PThread Count               8
Top K (Not Used)            40
Top P                       0.95
Temperature                 0.8
Repeat Penalty              1.4
Last token Count Penalty    64
Penalize New Line           True
Tfs Z                       1
Typical P                   1
Frequency Penalty           0
Prescence Penalty           0
Mirostat (Sampling Method)  2
Mirostat TAU                5
Mirostat ETA                0.1
```

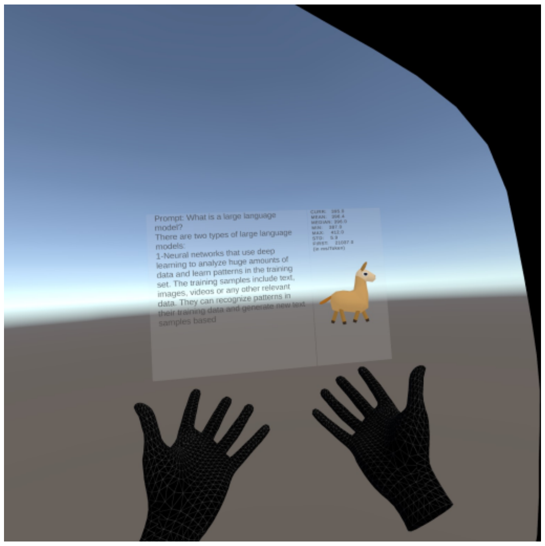
A full description of each parameter can be found at the llama.cpp repository [8].

## 4.4 Mobile VR/AR APPs

As a proof of concept we developed the scene to be compatible with Android and Meta Quest (AR/VR) deployment. The app is visible in figure 3.

## 5. CONCLUSION AND FUTURE WORK

In this study, we introduced a Unity-based multiplatform development tool designed for the research community. This tool aims to bridge the gap between Large Language Models like LLAMA2 and a variety of edge devices. In testing on the



```
Prompt: What is a large language
model?
There are two types of large language
models:
1-Neural networks that use deep
learning to analyze huge amounts of
data and learn patterns in the training
set. The training samples include text,
images, videos or any other relevant
data. They can recognize patterns in
their training data and generate new text
samples based
```

```
CURR:   395.9
MEAN:   396.4
MEDIAN: 396.0
MIN:    387.0
MAX:    412.0
STD:    5.9
FIRST:  21097.0
(in ms/Token)
```

**Figure 3: Running the application on the Meta Quest Pro**

Meta Quest Pro, we found that current inference times and delays make for a suboptimal user experience. Despite this, the tool's ability to quickly test and benchmark on different devices makes it valuable for iterative research and development. We also demonstrated that the 4-bit quantization appears to consistently give the best tradeoff in performance, most likely due to cache coherence on the architecture utilized.

## 5.1 Future Work

Our focus moving forward will be to refine the library, making it less tied to specific models. By integrating lower-level tensor frameworks such as ggml, we aim to enhance compatibility with a broader range of models within Unity. Additionally, there's potential to incorporate optimization tools that fine-tune deployments for specific platforms, especially targeting architectures like ARM64. Finally, we would like to provide more tools needed for edge device compilation as the current unity workflow can be challenging.

## ACKNOWLEDGMENTS

We thank the llama.cpp project and the broader open-source community for providing foundational tools that enabled our research. Special thanks to Bjorn Askew, Zander Brumbaugh, and Pedro Silva for their insights on our submission.

## REFERENCES

[1] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer, "Llm.int8(): 8-bit matrix multiplication for transformers at scale," 2022.

[2] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "Qlora: Efficient finetuning of quantized llms," 2023.

[3] T. Dettmers and L. Zettlemoyer, "The case for 4-bit precision: k-bit inference scaling laws," 2023.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[5] J. Durbin, "jondurbin/airoboros-l2-7b-2.1," https://huggingface.co/jondurbin/airoboros-l2-7b-2.1, 2023.

[6] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh, "GPTQ: Accurate post-training compression for generative pretrained transformers," *arXiv preprint arXiv:2210.17323*, 2022.

[7] G. Gerganov, "ggml," https://github.com/ggerganov/ggml, 2023.

[8] G. Gerganov, "llama.cpp," https://github.com/ggerganov/llama.cpp.git, 2023.

[9] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," 2017.

[10] T. Jobbins, "Thebloke/airoboros-l2-7b-2.1-gguf," https://huggingface.co/TheBloke/Airoboros-L2-7B-2.1-GGUF, 2023.

[11] D. Ranger, "llama.cpp-dotnet," https://github.com/dranger003/llama.cpp-dotnet, 2023.

[12] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. S. Koura, M.-A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. E. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, and T. Scialom, "Llama 2: Open foundation and fine-tuned chat models," 2023.

[13] Unity Technologies, "Unity." [Online]. Available: https://unity.com/

[14] J. Weizenbaum, "Eliza—a computer program for the study of natural language communication between man and machine," *Commun. ACM*, vol. 9, no. 1, p. 36–45, jan 1966. [Online]. Available: https://doi.org/10.1145/365153.365168