HEX-1 LOCAL DEPLOYMENT

Internship Technical Report

CPU-Only Offline Inference via llama.cpp (GGUF Q4_K_M)
BudEcosystem Hex-1 | June 2026 | DESKTOP-ASCP5AV

1 OBJECTIVE

The goal of this internship task was to run the **BudEcosystem Hex-1** language model — a 4-billion parameter model focused on Indian-language tasks — entirely offline on a low-resource Windows laptop, without GPU acceleration. Since the full-precision model exceeds the system's 8 GB RAM, a quantized GGUF version was produced and executed through llama.cpp's CPU-only inference engine.

2 TARGET SYSTEM SPECIFICATIONS

Component	Detail
Device name	DESKTOP-ASCP5AV
Processor	11th Gen Intel Core i3-1115G4 @ 3.00 GHz
RAM	8.00 GB installed (7.80 GB usable)
Graphics	Intel UHD Graphics (128 MB) — no dedicated GPU
Storage	330 GB used of 477 GB
System type	64-bit OS, x64-based processor

3 WORKING PRINCIPLE

3.1 What is Hex-1?

Hex-1 is a 4-billion parameter causal language model developed by BudEcosystem, trained with a focus on Indian multilingual tasks including Hindi, Tamil, Telugu, Kannada, Malayalam, and English. It follows a transformer decoder architecture similar to the LLaMA family, enabling text generation through next-token prediction.

3.2 Quantization (GGUF / Q4_K_M)

Running a 4B-parameter model at full precision (float32 or bfloat16) requires approximately 14–16 GB of RAM — far exceeding the available 8 GB. **Quantization** reduces this by representing model weights with fewer bits:

- Full precision (BF16): ~8 GB on disk, ~14 GB in memory

- Q4_K_M (4-bit mixed quantization): ~2.5 GB on disk, fits comfortably in 8 GB RAM

The **Q4_K_M** scheme uses 4-bit integers for most weights while keeping a small subset at higher precision for critical layers, balancing quality and compression. An **importance matrix (imat)** was also applied during conversion, which calibrates which weights matter most — improving output quality at the same compression ratio.

3.3 llama.cpp Inference Engine

llama.cpp is a C/C++ inference engine optimized for running quantized LLMs on consumer hardware. Key characteristics relevant to this deployment:

- Reads GGUF files natively without Python or PyTorch overhead
- Performs pure CPU matrix multiplication using BLAS-optimized routines
- Supports multi-threading — the 4 threads matched the i3-1115G4's thread count
- Maintains a KV cache in RAM to efficiently handle token context during generation

The inference loop works as follows: the input prompt is tokenized, each token is embedded, passed through the transformer decoder layers (attention + feed-forward), and the output logits are sampled to produce the next token — repeated until the completion is done.

4 SOLUTION APPROACH

Three options were evaluated before settling on a plan:

- **Full precision model (PyTorch/Transformers)** — rejected; the 4B-parameter model plus framework overhead would require well beyond 8 GB RAM.
- **Quantized GGUF via llama.cpp** — selected; Q4_K_M quantization shrinks the model to ~2.5 GB while remaining CPU-friendly.
- **Compiling llama.cpp from source** — abandoned after a missing CMake installation caused build errors; replaced with a pre-built binary instead.

5 STEP-BY-STEP IMPLEMENTATION

5.1 Downloading the Pre-Built llama.cpp Binary

From the official releases page (github.com/ggml-org/llama.cpp/releases), the CPU-only Windows x64 build was selected:

```
File: llama-b9651-bin-win-cpu-x64.zip  (16.1 MB)
```

Extracted to *C:\Users\Suresh\Downloads\llama-b9651-bin-win-cpu-x64*. Verified with:

```
.\llama-cli.exe --version
→ version: 9651 (e3bb1add8), built with Clang 20.1.8 for Windows x86_64
```

5.2 Installing Hugging Face CLI & Downloading the Model

```
pip install huggingface_hub
hf download budecosystem/hex-1 --local-dir C:\Users\Suresh\Downloads\hex1-model
```

This pulled the full-precision model (~8 GB across two safetensors shards) along with tokenizer and configuration files. A Hugging Face access token was created (Settings → Access Tokens, Write role) for authentication.

5.3 Converting to Quantized GGUF Format

The Hugging Face Space **gguf-my-repo** was used to perform the conversion remotely, avoiding the need for local Python dependencies:

Setting	Value Chosen
Source repo	budecosystem/hex-1
Quantization method	Q4_K_M
Importance matrix	Enabled (improves quality at same size)
Split model	Disabled (model small enough for one file)
Output repo	AANNAASSWWAARR/hex-1-Q4_K_M-GGUF

> *Note: An early attempt failed with a 404 error because the access token name was entered into the Hub Model ID field instead of the model identifier. Correcting to budecosystem/hex-1 resolved the issue.*

5.4 Running Inference

```
.\llama-cli.exe -m models\hex-1-q4_k_m-imat.gguf \
  -p "What is the capital of Kerala?" -n 200 --threads 4
```

Response from the model:

```
"The capital of Kerala is Thiruvananthapuram (also known as Trivandrum)."
Prompt: 3.1 t/s   |   Generation: 9.3 t/s
```

5.5 Desktop Launcher (Hex1.bat)

```
@echo off
cd C:\Users\Suresh\Downloads\llama-b9651-bin-win-cpu-x64
llama-cli.exe -m models\hex-1-q4_k_m-imat.gguf
```

Saved as **Hex1.bat** on the Desktop — allows launching the model with a double-click without typing commands each session.

5.6 Post-Setup Cleanup

Once all files were saved locally, the Hugging Face access token was deleted from account settings as a security best practice. llama.cpp runs entirely offline and requires no further network access or authentication.

6 MULTILINGUAL EVALUATION — MALAYALAM

> **Key Finding:** Hex-1 demonstrated *partial* multilingual capability. While the model was able to understand Malayalam text and perform Malayalam-to-English translation accurately, it struggled to generate fluent Malayalam responses. The generated output often contained mixed scripts and incorrect characters, indicating limited Malayalam text generation capability.

Observed Behaviour

Test Type	Outcome
Malayalam text comprehension	✓ Understood correctly
Malayalam → English translation	✓ Accurate
English → Malayalam generation	✗ Mixed scripts, garbled output
Pure Malayalam text generation	✗ Incorrect characters present

Analysis

This asymmetry is consistent with how multilingual models are trained: comprehension and translation tasks benefit from cross-lingual attention learned from parallel corpora, whereas *generation* in a script requires sufficient high-quality training examples in that script for the model to learn consistent tokenization and character mappings. The Q4_K_M quantization may also compress infrequently-used script tokens more aggressively, further degrading non-Latin text generation. Possible mitigations for future work include:

- Testing at a higher quantization level (Q5_K_M or Q6_K) to reduce character corruption
- Using a system prompt in English that instructs the model to respond in Malayalam
- Evaluating the full-precision model on a higher-RAM machine to isolate quantization effects

7 FINAL RESULT SUMMARY

Item	Outcome
Model	Hex-1 (4B params), Q4_K_M quantization, imat calibrated
Model file size	~2.5 GB (down from ~8 GB full precision)
Inference engine	llama.cpp, pre-built CPU-only Windows binary (build b9651)
Acceleration	None — pure CPU, 4 threads
Generation speed	~9.3 tokens/sec generation | ~3.1 tokens/sec prompt
Launch method	Desktop batch file (Hex1.bat) or PowerShell
English language tasks	Fully functional ✓
Malayalam comprehension	Functional ✓
Malayalam generation	Limited — mixed scripts observed ✗

Status	Deployed and verified offline

8 NOTES FOR FUTURE REFERENCE

- To run the model again, double-click **Hex1.bat** on the Desktop, or run llama-cli.exe with the **-m** flag pointing at the GGUF file.
- No internet connection or Hugging Face account is needed for day-to-day use — only the initial download and conversion required network access.
- For **more speed**: drop to Q2_K (trades quality for speed). For **better quality**: move to Q5_K_M or Q6_K (trades speed for quality).
- For **better Malayalam generation**: try Q5_K_M quantization, or use an English system prompt instructing the model to respond in Malayalam.
- The quantized GGUF repository remains available on Hugging Face at **AANNAASSWWAARR/hex-1-Q4_K_M-GGUF** for re-download if the local copy is lost.