Undergraduate Project Report

Quantization Workflows to Optimize Model Accuracy and Deployment Performance for LLMs

Submitted by:

Khushi Sahu
220532
khushis22@iitk.ac.in

Shikhar Chaurasia
221009
cshikhar22@iitk.ac.in

Supervisor:
Prof. R. M. Hegde

Department of Electrical Engineering
Indian Institute of Technology Kanpur

Contents

Abstract . . . 2
0.1 Introduction . . . 2
0.2 Problem Statement . . . 2
0.3 Objective . . . 3
0.4 Background: Quantization Concepts . . . 3
0.5 Quantization Techniques . . . 4
0.6 Multimodal Image–Text to Text Models . . . 5
0.7 Relative Size of Models . . . 5
0.8 Types of INT4 Quantization . . . 6
0.9 Types of INT8 Quantization . . . 7
0.10 Comparison of Vision-Language Models . . . 7
0.11 Evaluation Metrics . . . 7
0.12 COCO Dataset Evaluation . . . 10
0.13 Results . . . 11
0.14 Conclusion . . . 12

Abstract

Quantization is a key model compression technique that reduces floating-point weights and activations of neural networks into low-bit integer representations to enable efficient deployment on resource-constrained hardware. This report presents a complete study of quantization methods including INT8 and INT4 formats, block-wise quantization strategies used in `llama.cpp`, and the GGUF format for optimized CPU inference. It also covers multimodal image–text to text models, their training approaches, and evaluation metrics such as BLEU, METEOR, and ROUGE used for benchmarking on COCO dataset. This report consolidates theoretical, architectural, and evaluation aspects of quantization.

0.1 Introduction

Large Language Models (LLMs) and multimodal systems have grown exponentially in scale, often containing billions of parameters. Although these models achieve remarkable performance on tasks such as reasoning, image understanding, and text generation, deploying them in full FP32 or FP16 precision is impractical for most real-world hardware, including laptops, mobile devices, and embedded systems.

Quantization addresses these challenges by reducing the numerical precision of weights and activations, enabling up to 4–16$\times$ reductions in model size while greatly improving inference speed and memory efficiency. Importantly, modern quantization techniques preserve model accuracy to a large extent, making them suitable for both unimodal and multimodal AI tasks.

This project provides a detailed study of quantization, its need, techniques, applications in LLMs and multimodal models, and evaluation using standard metrics.

0.2 Problem Statement

Transformer architectures face several inherent limitations that restrict their practical deployment in real-world environments. A primary concern is their exceptionally high memory footprint; for example, a model such as LLaMA-70B requires more than 140 GB of memory in FP16 precision, far exceeding the capabilities of standard computing hardware. In addition, transformer inference is fundamentally memory-bound rather than compute-bound. The repeated retrieval of large weight matrices from RAM creates a significant latency bottleneck, as memory bandwidth becomes the dominant constraint limiting inference speed.

A further limitation arises from the restricted deployability of full-precision models. Running FP32 or FP16 LLMs on CPUs, mobile devices, or edge systems is largely infeasible due to their substantial memory and compute demands, necessitating the use of high-end GPUs equipped with large VRAM capacities. This requirement significantly reduces accessibility and prevents widespread usage of advanced LLMs. Quantization provides an effective solution to these challenges by reducing the numerical precision of model parameters, thereby lowering memory and bandwidth requirements while maintaining acceptable levels of model accuracy.

0.3 Objective

The objective of this project is to develop a clear understanding of quantization and its importance in enabling efficient deployment of large language models. This includes examining the theoretical basis of quantization, analyzing INT8, INT4, and blockwise methods used in modern LLMs such as LLaMA, and studying the GGUF model format designed for optimized inference. The project also aims to explore the architecture of multimodal image–text to text models and evaluate their performance using standard metrics such as BLEU, METEOR, and ROUGE, with a particular focus on the COCO benchmark methodology.

0.4 Background: Quantization Concepts

Quantization refers to the process of mapping continuous floating-point values to a reduced set of discrete integer levels. In deep learning, this process is commonly applied to model weights, which constitute the majority of a network's memory footprint, and can also be extended to activations and, in some cases, gradients. By lowering numerical precision, quantization reduces storage requirements and enables more efficient computation during inference.

Memory Requirements

The memory reduction achieved through quantization is significant. A model with N parameters requires:

$$\begin{aligned} \text{FP32:}&\ 4N \text{ bytes},\\ \text{FP16/BF16:}&\ 2N \text{ bytes},\\ \text{INT8:}&\ N \text{ bytes},\\ \text{INT4:}&\ 0.5N \text{ bytes}. \end{aligned}$$

For example, a 70-billion-parameter model in FP16 precision occupies approximately 140 GB of memory, making full-precision deployment infeasible on most systems. Quantization is therefore essential for enabling the practical use of large language models in resource-constrained environments.

0.5 Quantization Techniques

Quantization can be broadly categorized into:

- **Post-Training Quantization (PTQ):** Weights are quantized after training, making this approach simple and computationally efficient.
- **Dynamic Quantization:** Activations are quantized during inference, reducing compute overhead without requiring calibration data.
- **Static Quantization:** Calibration data is used to estimate activation ranges prior to inference, improving quantization accuracy.
- **Blockwise Quantization (LLaMA):** LLaMA models employ fine-grained blockwise quantization:
    1. Split weights into blocks of 32 or 64 elements.
    2. Compute a scale value for each block.
    3. Normalize the block values.
    4. Quantize values into INT2–INT8.
    5. Store quantized values along with scales and optional zero-points.

This approach significantly reduces quantization error compared to coarse tensor-level quantization.

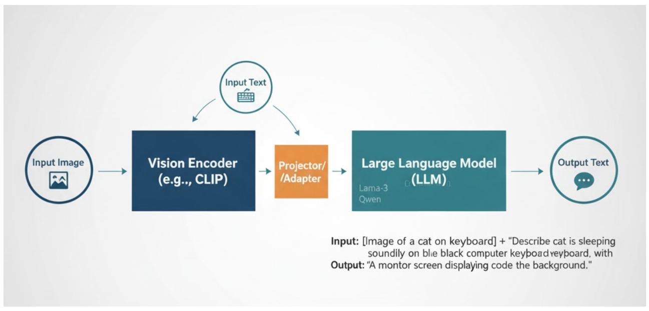
GGUF Format: The GGUF format used by `llama.cpp` stores:

- Tokenizer vocabulary,
- Model hyperparameters,
- Quantized weight tensors,
- Per-block scales and zero-points,
- KV-cache configuration.

GGUF models load quickly, run efficiently on CPU and Metal/Vulkan backends, and require minimal RAM.

0.6 Multimodal Image–Text to Text Models

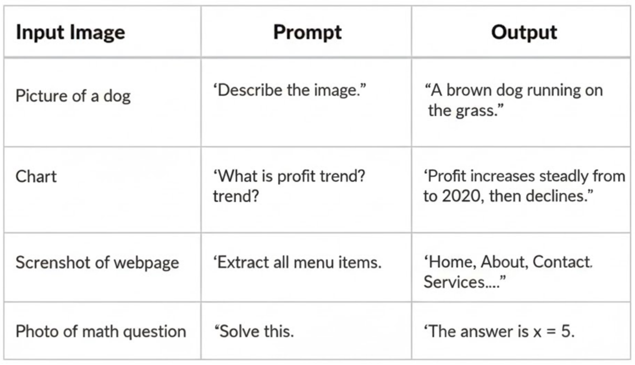
Multimodal models integrate information from both images and text to generate coherent textual responses. They typically consist of a vision encoder, which extracts features from the image; a fusion mechanism that aligns visual and textual representations; and a language model decoder that produces the final output. By combining these components, such models can interpret visual content and relate it to linguistic context.



Input: [Image of a cat on keyboard] + "Describe cat is sleeping soundily on blie black computer keyboad veyboard, with
Output: "A montor screen displaying code the background."

Figure 1: Working of vision model.

Training multimodal systems involves several approaches, including contrastive learning for aligning modalities, instruction tuning for improved prompt following, caption-generation training, and visual question answering datasets to enhance reasoning. Models such as LLaVA, MiniCPM, Qwen-VL, and Mistral-Vision exemplify these techniques and are widely used for tasks such as captioning, multimodal reasoning, and interactive visual–language communication.



Input Image	Prompt	Output
Picture of a dog	'Describe the image."	"A brown dog running on the grass."
Chart	'What is profit trend? trend?	'Profit increases steadly from to 2020, then declines."
Screnshot of webpage	'Extract all menu items.	'Home, About, Contact. Services...."
Photo of math question	'Solve this.	'The answer is x = 5.

Figure 2: Examples for Image-Text to Text model.

0.7 Relative Size of Models

- **FP32** Highest precision representation; used primarily in training but rarely in inference due to high memory and compute cost.

- **FP16** Commonly used in mixed-precision training; more efficient than FP32 but limited by a smaller dynamic range.
- **BF16** Provides a larger dynamic range than FP16 and is widely supported by Google TPUs; preferred for stable training.
- **INT8** Enables efficient 8-bit matrix multiplications; offers a strong trade-off between accuracy and computational efficiency.
- **INT4** Represents weights in 4 bits; often only weights are quantized while activations remain in higher precision; highly beneficial for large models.
- **INT2** Extremely low-precision representation used in microcontroller and ultra-low-memory environments.

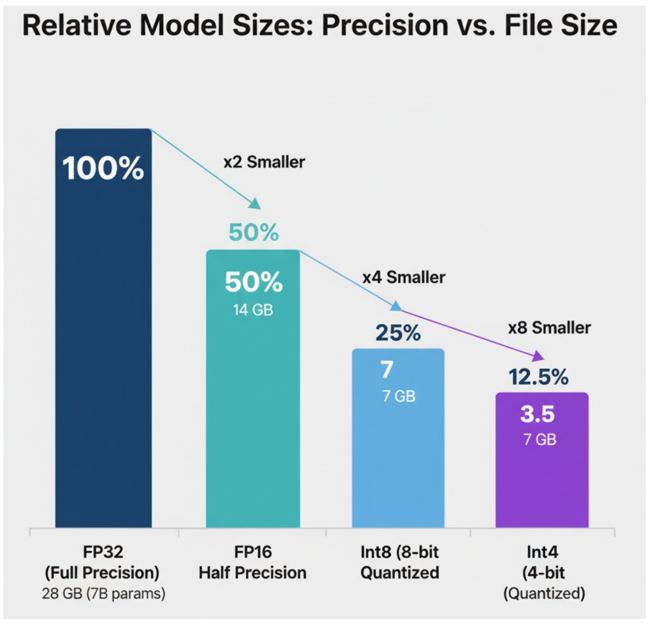


Figure 3: Relative model sizes with different quantizations

0.8 Types of INT4 Quantization

- **Q4_0** Baseline INT4 quantization using per-block scaling without a zero-point; simplest but generally lower accuracy.
- **Q4_1** Incorporates a zero-point for improved expressiveness and typically yields better accuracy than Q4_0.
- **Q4_K_M** A K-quant (group-wise) method using optimized block structures; provides high stability and the best overall INT4 quality.

- **Q4_K_S** Symmetric K-quant variant designed for stable behavior when weights are symmetrically distributed.

0.9 Types of INT8 Quantization

- **Q8_0** Basic and accurate INT8 quantization; uses per-block scaling without zero-points and is widely deployed.
- **Q8_K** K-quant optimized INT8 format with group-wise scaling; reduces quantization error and is notably effective for large LLaMA models.
- **Q8_S** Symmetric INT8 variant that simplifies computation and performs well when weights are centered around zero.

0.10 Comparison of Vision-Language Models

It is important to compare the major vision–language models that are commonly deployed using INT4 & INT8 formats. Models such as Qwen, LLaVA, and MiniCPM vary significantly in terms of parameter size, reasoning capability, efficiency, and quantization friendliness. The following table summarizes their characteristics and ideal use-cases under quantized settings.

Model	**Params**	**Strengths**	**Weaknesses**	**Ideal For**
Qwen	3B, 7B	Strong reasoning, coding capability, multilingual support	Heavy at 7B; slower than lightweight models	Reasoning tasks, coding, multilingual inference
LLaVA	7B, 8B	Strong VLM alignment, good captioning and VQA	Weaker long-form reasoning	Captioning, multimodal chat, VQA
MiniCPM	4B, 8B	Efficient, lightweight, quantization-friendly	Not strong in math or deep reasoning	Edge deployment, quantized inference

Table 1: Comparison of Vision–Language Models.

0.11 Evaluation Metrics

- **BLEU (Bilingual Evaluation Understudy)**

BLEU measures how closely a generated caption matches one or more reference captions by evaluating the overlap of n-grams. It computes modified n-gram precision, which reflects how many n-grams in the candidate appear in the references, while avoiding inflated counts from repeated words. The n-gram precision is defined as:

$$P_n = \frac{\text{Number of matched } n\text{-grams}}{\text{Total number of } n\text{-grams in generated sentence}}.$$

To combine the precisions across different n-gram sizes, BLEU uses the geometric mean:

$$GM = \exp\left(\sum_{n=1}^{N} w_n \log P_n\right),$$

where typically $N = 4$ and each weight is $w_n = \frac{1}{4}$.

BLEU also applies a brevity penalty (BP) to penalize overly short generated sentences. The penalty depends on the candidate length c and the reference length r:

$$BP = \begin{cases} 1, & c > r, \\ e^{1-\frac{r}{c}}, & c \leq r. \end{cases}$$

The final BLEU score is obtained by combining the brevity penalty with the geometric mean:

$$\text{BLEU} = BP \times GM.$$

- **METEOR**

METEOR is designed to align more closely with human judgment by evaluating semantic similarity rather than relying purely on surface n-gram overlap. Unlike BLEU, METEOR performs unigram-level matching using multiple linguistic strategies, including exact matches, stemmed matches, synonym matches, and paraphrase matches. This enables METEOR to capture a broader range of valid linguistic variations in generated captions.

Step 1 — Unigram Matching: Words from the generated caption are matched to reference-caption words through:

- Exact match,
- Stem match,
- Synonym match,
- Paraphrase match.

Step 2 — Compute Precision and Recall: Once matches are found, unigram precision and recall are computed as:

$$P = \frac{\text{matched unigrams}}{\text{candidate unigrams}}, \qquad R = \frac{\text{matched unigrams}}{\text{reference unigrams}}.$$

Step 3 — Weighted F-Score: METEOR combines precision and recall using a weighted harmonic mean, giving recall nine times more weight than precision to encourage completeness of generated content:

$$F_{\text{mean}} = \frac{10PR}{R + 9P}.$$

Step 4 — Fragmentation Penalty: METEOR penalizes scattered or disordered matches by computing a penalty based on the number of contiguous matched chunks:

$$\text{Penalty} = 0.5 \left(\frac{\text{chunks}}{\text{matches}} \right)^3.$$

Step 5 — Final METEOR Score: The final score incorporates the penalty to adjust the weighted F-score:

$$\text{METEOR} = F_{\text{mean}} \times (1 - \text{Penalty}).$$

METEOR's combination of semantic matching, weighted recall, and fragmentation penalty makes it an effective metric for evaluating linguistic richness, semantic adequacy, and sentence coherence in generated captions.

- **ROUGE (Recall-Oriented Understudy for Gisting Evaluation)**

ROUGE is a recall-focused evaluation metric widely used in summarization and descriptive text generation. It measures how much of the reference content is captured by the generated output. ROUGE-N computes recall over matching n-grams between the reference and the candidate, defined as:

$$\text{ROUGE-N} = \frac{\sum_{\text{gram}_n \in \text{ref}} \min(\text{count}_{\text{gen}}, \text{count}_{\text{ref}})}{\sum_{\text{gram}_n \in \text{ref}} \text{count}_{\text{ref}}}.$$

ROUGE-L, on the other hand, evaluates sentence-level structure by using the longest common subsequence (LCS), which is the longest sequence of words appearing in both the candidate and reference in the same order.

ROUGE-L Recall:

$$R_{LCS} = \frac{LCS}{\text{length of reference}}.$$

ROUGE-L Precision:

$$P_{LCS} = \frac{LCS}{\text{length of candidate}}.$$

ROUGE-L F-Score:

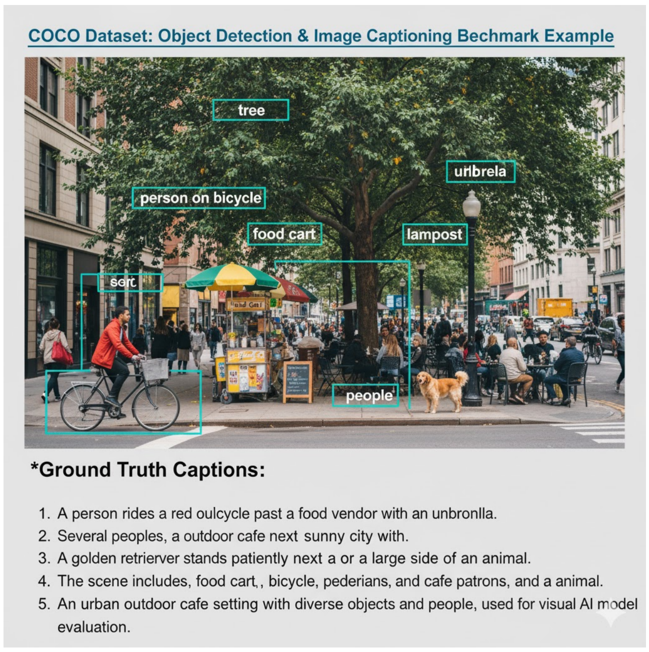
$$F_{LCS} = \frac{(1+\beta^2)\, R_{LCS}\, P_{LCS}}{R_{LCS} + \beta^2 P_{LCS}},$$

where $\beta = 1.2$ is typically used to weight recall slightly more than precision.

ROUGE thus provides a measure of completeness and structural fidelity and serves as a complementary metric to BLEU and METEOR by emphasizing coverage of reference content and sentence-level similarity.

0.12 COCO Dataset Evaluation

The COCO dataset contains over 330,000 images, each accompanied by five human-written captions, making it a widely used benchmark for evaluating image captioning models. Its multi-caption structure provides a reliable basis for assessing the descriptive quality and variability of generated outputs.


COCO Dataset: Object Detection & Image Captioning Bechmark Example
tree
unbrela
person on bicycle
food cart
lampost
sort
people

***Ground Truth Captions:**

1. A person rides a red oulcycle past a food vendor with an unbronlla.
2. Several peoples, a outdoor cafe next sunny city with.
3. A golden retrierver stands patiently next a or a large side of an animal.
4. The scene includes, food cart,, bicycle, pederians, and cafe patrons, and a animal.
5. An urban outdoor cafe setting with diverse objects and people, used for visual AI model evaluation.

Figure 4: example from COCO dataset

Evaluation involves generating a caption for 100 images, comparing it with the corresponding reference captions, and computing metrics such as BLEU, METEOR, and ROUGE. The results are then aggregated over the entire dataset.

Models including LLaVA, Qwen2.5, MiniCPM, and Mistral have been evaluated using this standardized methodology.

0.13 Results

The performance of the evaluated vision–language models was measured under INT8 and INT4 quantization settings using 100 COCO images. Models were compared on the basis of BLEU, METEOR, and ROUGE scores, along with average inference latency and quantized model size. These measurements provide a direct view of how quantization affects accuracy, efficiency, and deployability across different architectures.

Model	**Params**	**Quant.**	**Size**	**Latency**	**BLEU**	**METEOR**	**ROUGE**
Qwen 2.5	7B	Q8_0	8.10 GB	8.289	0.0118	0.2202	0.1207
Qwen 2.5	3B	Q8_0	3.29 GB	5.084	0.0007	0.1491	0.0521
MiniCPM V4.5	8B	Q8_0	8.71 GB	22.07	0.0124	0.1659	0.1479
MiniCPM V4	4B	Q8_0	3.83 GB	5.834	0.0136	0.1969	0.1423
LLaVA 1.6	7B	Q8_0	7.70 GB	24.33	0.0162	0.2113	0.1235
LLaVA 3	8B	Q8_0	8.54 GB	4.598	0.0428	0.1833	0.5056

Table 2: INT8 Quantization Performance Across Models

Model	**Params**	**Quant.**	**Size**	**Latency**	**BLEU**	**METEOR**	**ROUGE**
Qwen 2.5	7B	BF16	15.2 GB	14.276	0.0174	0.2277	0.1265
Qwen 2.5	7B	Q4_K_M	4.68 GB	3.544	0.0099	0.2209	0.1215
Qwen 2.5	3B	BF16	6.18 GB	8.136	0.0009	0.1473	0.0531
Qwen 2.5	3B	Q4_K_M	1.93 GB	2.527	0.0008	0.1429	0.0494
MiniCPM V4.5	8B	Q4_K_M	5.03 GB	7.458	0.0036	0.1134	0.1536
MiniCPM V4	4B	Q4_K_M	2.19 GB	4.923	0.0103	0.1276	0.1423
LLaVA 1.6	7B	Q4_K_M	4.37 GB	12.802	0.0136	0.2066	0.0785
LLaVA 3	8B	Q4_K_M	4.92 GB	2.690	0.4170	0.2667	0.1853

Table 3: INT4 Quantization Performance Across Models

Overall, INT8 quantization preserves accuracy well across all models, with Qwen 2.5 (7B) and LLaVA 1.6 achieving the strongest METEOR and ROUGE performance. INT4 quantization significantly reduces model size and latency, making models like MiniCPM 4B and Qwen 3B highly suitable for edge deployment. LLaVA 3 (8B) shows exceptionally strong BLEU and METEOR under INT4, demonstrating that fine-grained Q4_K_M quantization retains high captioning quality while offering large memory savings.

0.14 Conclusion

Quantization is a crucial technique for enabling the deployment of large language models and multimodal systems on standard hardware. Methods such as INT8 and INT4 quantization, along with fine-grained blockwise approaches, allow substantial reductions in memory and computational requirements while preserving model accuracy. The GGUF format further enhances inference efficiency on CPU and mobile platforms. Evaluations using BLEU, METEOR, and ROUGE on the COCO dataset demonstrate that quantized models continue to deliver competitive performance. This UGP project provides a consolidated examination of the fundamental principles and practical applications of quantization in modern AI systems.