Georg-August-Universität Göttingen

Faculty of Mathematics and Computer Science

# Master's Thesis

# Optimizing LLM Deployment via Cross-Precision Transfer: A Case Study in Biomedical AI Agent Biomni

**Hasan Marwan Mahmood Aldhahi**

Supervised by:
Prof. Julian Kunkel
Dr. Narges Lux

February 2026

# Abstract

Running Large Language Models (LLMs) in hardware-constrained settings requires ever-greater levels of aggressive compression; in particular, 8-bit floating-point (FP8) is now emerging as a standard for reducing memory costs. Yet despite the rapid adoption of 8-bit quantization in production, a large technical gap remains: domain-specific adaptations are always assembled in FP16/BF16 precision, creating a precision mismatch whenever those patches are applied over a pre-quantized FP8 base. This thesis seeks to answer whether high-precision Low-Rank Adaptation (LoRA) modules can be successfully 'transplanted' onto FP8 base models without retraining, and thus be used on existing shared systems to run domain-specific biomedical adapters.

We designed and prototyped three approaches: (1) Naive Transfer, (2) Corrective Extraction, and (3) Direct Quantization. Testing all three on the Eval1 biomedical benchmark—a challenging test set of 433 questions on CRISPR, GWAS, and rare disease diagnostics—we obtained surprising results: despite the expectation that accuracy should suffer under precision loss, Naive Transfer actually achieved 44.6% accuracy, matching our 44.5% full-precision baseline.

This evidence supports the *Orthogonality Hypothesis*: the observation that the effect of quantization noise (high-rank, isotropic) is orthogonal to the semantic information encoded in LoRA (low-rank, structured). A key finding was the near-zero cosine similarity on average across all model layers ($\approx -0.00001$) between the LoRA adapter vector and the quantization noise vector, and it turns out this high-dimensional weight space is able to accommodate both parts without functional interference.

Practical implications include a scalable way to enable resource-efficient AI on existing infrastructure. An extension to vLLM's multi-adapter system allows us to incorporate our specialized biomedical tools like Biomni into existing Chat AI infrastructure, requiring no additional GPU cost, by simply loading the LoRA adapter alongside the pre-quantized Qwen-32B-FP8 base model. This allows domain-specific AI services to share GPU resources with general-purpose models, reducing the hardware and operational costs of running high-throughput biomedical AI services.

# Declaration on the use of ChatGPT and comparable tools

In this work I have used ChatGPT or another AI as follows:

| | |
|---|---|
| ☐ | Not at all |
| ☑ | During brainstorming |
| ☐ | When creating the outline |
| ☑ | To write individual passages, altogether to the extent of 20% of the entire text |
| ☐ | For the development of software source texts |
| ☐ | For optimizing or restructuring software source texts |
| ☑ | For proofreading or optimizing |
| ☐ | Further, namely: – |

I hereby declare that I have stated all uses completely.
Missing or incorrect information will be considered as an attempt to cheat.

Hasan Marwan Mahmood Aldhahi February 2026

# Acknowledgements

I thank my supervisors, Prof. Julian Kunkel and Dr. Narges Lux, for their help and support during this work. Their comments and advice influenced the design of the study and the presentation of the results, especially in the areas of systems optimization and machine learning. I also thank the open source community for the tools used in this project. I am grateful to the teams behind vLLM, MergeKit, `llm-compressor`, and the Transformers library for the software that this work depends on. The open-source community's commitment to transparency and reproducibility continues to advance the field of machine learning.

Special thanks to the creators of the Qwen and Biomni models for making their work publicly available, enabling research on quantization and domain adaptation.

Finally, I would like to thank my family and friends for their unwavering support and encouragement during the course of this Master's program.

# Contents

# List of Figures

# List of Tables

# Listings

# Chapter 1

# Introduction

## 1.1 Motivation

The scale of LLMs has grown significantly, leading to a persistent bottleneck and the need for more hardware infrastructure, with GPU Memory (VRAM) serving as the key limitation. Despite the impressive reasoning power of modern model architectures like Qwen3-32B [1], deploying such models in native Brain Float (BF16) precision remains expensive or technically unfeasible for many service providers. For example, a 32-billion parameter model requires approximately 64 GB of VRAM in BF16; this threshold is unrealistic for consumer-grade hardware and imposes a high cost of entry for production environments. A more scale-efficient solution to this memory crisis is Post-Training Quantization (PTQ) to FP8 [2], which reduces hardware requirements by half by scaling weight and activation precision from 16 bits down to 8 while maintaining almost the same performance as the full-precision baseline. This transition is facilitated by the native hardware support provided in contemporary NVIDIA architectures, such as the Hopper and Ada Lovelace series.

The real difficulty, however, arises when we move beyond general-purpose models. Domain-specific models are usually either all the weights fully fine-tuned on domain specific datasets or small fraction of newly added layers (LoRA adapters) are fine-tuned [3]. The latter approach is almost always conducted in high-precision (BF16 or FP16) to ensure numerical stability between the original frozen weights and the learned adapter layer weights. This creates a dilemma: a researcher might have a high-precision biomedical adapter but only have the VRAM capacity to host an FP8 base model. Currently, there is no standardized way to combine these two without the high cost of merging the adapter with the base model, then re-quantizing the entire system or retraining the adapters to accommodate the FP8 base model's quantization error and preserve performance

### 1.1.1 The Resource-Efficient Deployment Challenge

This research grew out of a practical challenge at the Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen (GWDG). The existing Chat AI service already hosts Qwen3-32B-FP8 [1] on its shared infrastructure for general tasks. Our primary goal was to find a way to integrate specialized agentic biomedical capabilities (Biomni R0-32B) [4] into this existing framework without the need to use extra GPUs.

If we simply hosted a separate, full-precision Biomni R0-32B model, it would require 64 GB of VRAM, doubling our infrastructure costs. Alternatively, hosting an FP8-quantized version of the fully fine-tuned BF16 Biomni R0-32B model would still require an additional 32 GB of VRAM. Instead, we sought to exploit vLLM's [5] multi-adapter capabilities by dynamically loading a LoRA adapter [3] onto the pre-existing FP8 base model. This approach would enable specialized biomedical reasoning through simple endpoint switching, so multiple task-specific variants could share the same underlying GPU resources. At the same time, it would preserve the ability to serve the standard Qwen-32B-FP8 base model [1] for general-purpose use.

## 1.2 Problem Statement

Broadly, this work asks whether high-precision domain adaptations can be transferred to an FP8-quantized base model without materially degrading performance on domain-specific tasks. This question is especially relevant for biomedical NLP, where models such as Biomni-R0-32B must answer complex queries that require multi-step reasoning, including genetic variant interpretation and GWAS analysis (genome-wide association studies that link genetic variants to diseases or traits). In practice, however, naively applying a BF16 LoRA adapter on top of an FP8 base model raises several challenges, from mixed-precision implementation details to more fundamental concerns about interference in weight space when combining quantized and high-precision updates. Specifically, we must investigate whether the noise introduced by the FP8 quantization process will 'drown out' or corrupt the specialized biomedical knowledge contained within the LoRA adapter without the need to retrain the adapter.

## 1.3 Research Questions

This thesis investigates the following research questions:

1. **RQ1: Feasibility of Cross-Precision Transfer**
   How much performance can be maintained without retraining when extracted BF16 LoRA adapters from the fine-tuned domain-specific model and the base model are applied to FP8-quantized base models?
2. **RQ2: Orthogonality of Quantization Noise and Semantic Adaptation**
   Do domain-specific knowledge and quantization error noise occupy geometrically orthogonal subspaces in the weight embedding space, preventing substantial mutual interference?
3. **RQ3: Corrective vs. Naive Transfer**
   Is it feasible to design BF16 LoRA adapter with a certain rank that simultaneously mitigates quantization noise from FP8 quantization and encodes domain knowledge without the need to retrain the adapter, or does this dual-objective approach inevitably lead to capacity saturation in representational power for accommodating the noise and representing the domain knowledge?
4. **RQ4: Resource-Efficient Deployment**
   Can cross-precision transfer facilitate the long-term, resource-efficient rollout of specialized LLMs on shared infrastructure without necessitating further hardware investment for specialized agentic models?

## 1.4 Research Contributions

This thesis presents three primary contributions to the field of efficient LLM deployment:

1. **Empirical Validation of the Orthogonality Hypothesis**
   Naive Transfer is proposed as a method to potentially achieve high performance retention, which is aimed to match the full-precision baseline. We analyze the weight-space geometry to determine if a near-zero cosine similarity on average across all layers exists between the extracted BF16 LoRA adapter and the quantization noise vectors. Such result would support the hypothesis that low-rank semantic knowledge is geometrically separated from high-rank quantization noise.

2. **Comparative Analysis of Transfer Methods**
   We systematically compare three approaches to cross-precision transfer:

   - **Method A (Naive Transfer)**: Extracting LoRA from the difference between fine-tuned and base models in BF16, then applying to the FP8 base of the same family of model. This method is evaluated to determine its robustness compared to retraining approaches in which LoRA adapters normally are used to be fine-tuned from scratch.
   - **Method B (Corrective Extraction)**: Extracting LoRA from the difference between fine-tuned BF16 and dequantized FP8 models since the dequantized weights are used during inference of quantized models in production. This method is evaluated to determine if it suffers from quantization noise.
   - **Method C (Direct Quantization)**: Quantizing the domain-adapted model directly using FP8 with domain-specific calibration data. This serves as our quantized baseline for the comparison.

3. **Resource-Efficient Deployment Architecture**
   We demonstrate a practical deployment strategy using vLLM's multi-adapter capability. By dynamically loading a LoRA adapter onto an existing generic FP8 base model, agentic biomedical capabilities can be provided without the need for additional GPU resources for hosting. Consequently, this approach would significantly reduce VRAM requirements compared to hosting separate specialized quantized or full-precision models, with efficient endpoint switching between generic and domain-specific model capabilities from the same model family, such as Qwen-32B.

## 1.5 Thesis Structure

The remainder of this thesis is organized as follows:

**Chapter 2: Background**
Introduces the challenge of deploying 32B-scale LLMs under memory constraints. Presents LoRA and post-training quantization (FP8, block-wise) as key techniques. Defines the Biomni project and Eval1 benchmark. Introduces the Orthogonality Hypothesis as a theoretical basis for LoRA transfer.

**Chapter 3: Methodology**
Evaluates three LoRA transfer methods: Naive Transfer, Corrective Extraction, and Direct Quantization. Describes the experimental setup using Biomni-R0-32B and Eval1. Details implementation choices (arithmetic, dequantization, vLLM integration) to ensure reproducibility.

**Chapter 4: Results**
Presents performance results across Eval1, including task-level breakdowns. Compares methods against Qwen3-32B-BF16 and Biomni-R0-32B-INT8. Analyzes performance vs. LoRA rank and evaluates memory footprint.

**Chapter 5: Discussion**
Explains limitations of Corrective Extraction due to quantization error. Revisits Naive Transfer via the Orthogonality Hypothesis. Discusses broader implications for efficient, resource-efficient AI deployment.

**Chapter 6: Conclusion**
Summarizes the key findings and the practical impact of cross-precision transfer. Highlights primary contributions, including a recommended deployment workflow and infrastructure sharing strategies. Outlines future research directions such as multi-adapter scenarios, generalization to other quantization formats, and the exploration of higher-rank adapters.

# Chapter 2

# Background

This chapter covers the background for studying LoRA updates across different numerical precisions. It starts with large language models and why memory use becomes a limiting factor at scale. It then describes post training quantization, focusing on FP8 and block wise approaches. Next, it explains Low Rank Adaptation as a practical method for fine tuning. The chapter also states the Orthogonality Hypothesis in mathematical form and summarizes the Biomni project and the Eval1 benchmark used in the experiments.

## 2.1 Large Language Models and Memory Constraints

Transformer based large language models [6] achieve strong results on many natural language understanding tasks, including reasoning and problem solving. Models such as GPT-3 [7], LLaMA [8], and Qwen [1] rely on very large parameter counts, ranging from billions to trillions, which provide the capacity behind their performance. Consequently, the hardware overhead for hosting these systems is extraordinarily high. Given a model with $N$ parameters held in $b$-bit precision, the baseline memory demand stands at:

$$M_{model} = N \times \frac{b}{8} \text{ bytes} \tag{2.1}$$

For Qwen3-32B in BF16 (16-bit) precision:

$$M_{model} = 32 \times 10^9 \times \frac{16}{8} = 64 \text{ GB} \tag{2.2}$$

This exceeds the capacity of consumer GPUs with 8-24 GB of VRAM and requires expensive datacenter hardware. During inference, memory must be allocated for activations known as the KV cache. This memory is used for attention mechanisms and intermediate computations for efficient computation. Since the model is computing the next tokens based on the previous tokens in auto-regressive manner, the KV cache is used for keeping track of the previous tokens both key and value tensors to avoid recomputing them at each new token to be generated. KV cache increases further the total memory footprint. The solution to this problem is to use Post-Training Quantization (PTQ) [9] to reduce the bit precision of model weights and activations to be able to free more VRAM to be used in such overhead of KV cache and as well as increasing batch processing and sequence length, boosting throughput and lowering latency for token generation during inference.

## 2.2 Post-Training Quantization

Post-Training Quantization (PTQ) reduces the bit precision of model weights and activations without demanding a retrain [9]. Unlike Quantization-Aware Training (QAT), which blends quantization into the training loop, PTQ works on ready-made models, rendering it more viable for massive models where retraining is too costly or time-consuming. It requires a calibration dataset for determining the scale and bias parameters for the quantization. Most of the frameworks for PTQ require a calibration dataset which is taken as industry standard [9]. Even though in the documentation it says it is not required to provide a calibration dataset, our domain specific application is specific and sticking to the industry standard is more robust and reliable. In the next subsections, we will discuss these concepts thoroughly on how PTQ FP8 quantization works and most notably Block-wise Quantization which is the one used for the Qwen3-32B model to quantize it into FP8 precision from the original authors of Alibaba Cloud.

### 2.2.1 FP8 Floating-Point Format

The IEEE FP8 standard [2] defines two 8-bit floating-point formats:

- **E4M3** (4 exponent bits, 3 mantissa bits): Provides higher precision but limited dynamic range, typically used for weights
- **E5M2** (5 exponent bits, 2 mantissa bits): Provides wider dynamic range but lower precision, typically used for activations and gradients

The E4M3 format values range is $[-448, 448]$ with varying precision depending on magnitude. Modern NVIDIA GPUs with compute capabilities of +8.9, like Hopper H100 and Ada Lovelace, provide native hardware acceleration for FP8 arithmetic, making it not only memory-efficient but also computationally fast.

### 2.2.2 Block-wise Quantization

Naive per-tensor quantization applies a single scale factor to all elements of a tensor:

$$W_{quantized} = \text{round}\left(\frac{W_{FP16}}{s}\right) \tag{2.3}$$

where $s$ is a scalar scale factor. However, this approach performs poorly when weights have heterogeneous magnitudes across different regions of the tensor.

Block-wise quantization [10] divides tensors into blocks and applies separate scale factors to each block:

$$W_{quantized}[i, j] = \text{round}\left(\frac{W_{FP16}[i, j]}{s[b_i, b_j]}\right) \tag{2.4}$$

where $b_i, b_j$ are the block indices corresponding to position $(i, j)$. For a weight matrix of shape $(H, W)$ with block size $B$, the scale tensor has shape $(\lceil H/B \rceil, \lceil W/B \rceil)$.

The Qwen-FP8 implementation uses Block-128 quantization (also known as fine-grained FP8 quantization), meaning each block contains 128 elements along the quantization dimension. This block size is documented in the official Qwen3 model card and provides a good balance between quantization accuracy and metadata overhead.

### 2.2.3 Dequantization

To perform arithmetic operations on quantized weights in higher precision, dequantization is required:

$$W_{FP16} = W_{FP8} \times s_{expanded} \tag{2.5}$$

where $s_{expanded}$ is the scale tensor expanded to match the shape of $W_{FP8}$ by repeating each scale value along the appropriate dimension.

## 2.3 Low-Rank Adaptation (LoRA)

Low-Rank Adaptation [3] is a parameter-efficient fine-tuning technique that freezes the pre-trained model weights and injects trainable low-rank decomposition matrices into each layer.

### 2.3.1 Mathematical Formulation

For a pre-trained weight matrix $W_0 \in \mathbb{R}^{d\times k}$, LoRA represents the weight update as:

$$W = W_0 + \Delta W = W_0 + BA \tag{2.6}$$

where $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times k}$ with rank $r \ll \min(d, k)$. During training, $W_0$ is frozen and only $B$ and $A$ are updated.

The number of trainable parameters is:

$$\text{Params}_{LoRA} = r \times (d + k) \tag{2.7}$$

For typical values ($r = 256$, $d = 5120$, $k = 5120$), this represents only $\sim$5% of the original parameters.

### 2.3.2 Connection to Singular Value Decomposition

LoRA can be interpreted through the lens of Singular Value Decomposition (SVD). The full weight update $\Delta W$ can be decomposed as:

$$\Delta W = U\Sigma V^T \tag{2.8}$$

where $\Sigma$ is a diagonal matrix of singular values. LoRA approximates this by keeping only the top $r$ singular values:

$$\Delta W \approx U_r \Sigma_r V_r^T \tag{2.9}$$

This low-rank approximation is effective because task-specific adaptations often lie in a low-dimensional subspace [11].

### 2.3.3 Rank Selection and Capacity

The choice of rank $r$ represents a trade-off between expressiveness and parameter efficiency. Higher ranks can capture more complex adaptations but require more memory and computation. Empirically, ranks between 64 and 512 are commonly used for LLMs.

Critically, the rank determines the *capacity* of the adapter. If the adaptation task requires encoding information that cannot be compressed into rank $r$, the adapter will fail to learn effectively.

## 2.4 The Biomni Project

Biomni is a general-purpose AI agent platform developed at Stanford University with the goal of automating and accelerating biomedical research workflows across a broad range of subfields [12]. It is designed to operate as a 'virtual AI biologist' capable not only of answering questions, but also of designing, executing, and coordinating complex multi-step workflows that would normally require the guidance of expert human scientists. As opposed to conventional systems based on rigid templates, Biomni generates highly adaptable sequences of actions across genomics, transcriptomics, clinical informatics, and other fields. This is achieved by combining Large Language Model (LLM) reasoning with a sophisticated set of information retrieval and code execution tools to call upon software routines, build workflows, and interpret results dynamically. The innovation behind the Biomni is its agentic environment, Biomni-E1, in which over 150 specialized tools, 105 software packages, and 59 databases are directly accessible to the agent to support research problem solving and task automation. The system integrates several tools for prediction, dataset retrieval, sequence and structure checks, and experiment planning. It can also generate and execute Python code. Since it connects to resources used in genomics, pharmacology, and systems biology, it can be applied to a range of projects. In practice, it relies on established databases such as AlphaFold protein structures, NCBI related to sequence data and alignment, and the GWAS Catalog which links between genetic variants and traits. The mixture of integrated resources and flexible agentic planning enables Biomni to carry out sophisticated biomedical analyses that would otherwise need extensive expert manual planning and coordination. The source model studied in this research, Biomni-R0-32B, is built upon this agentic platform. It is a large language model based on the Qwen3-32B architecture, further adapted and fine-tuned for biomedical reasoning. (Details of the model architecture and experimental evaluation follow below.)

Table 2.1 presents a sample of high-impact tools available in Biomni-E1, each serving essential roles in biomedicine:

Table 2.1: **Selected High-Impact Biomni Tools**

| **Tool Name** | **Category** | **Primary Function** | **Notes / Cost** |
|---|---|---|---|
| `advanced_web_search_claude` | **Web / Research** | Performs complex, reasoning-based web searches to retrieve up-to-date scientific information and verify facts using Claude models. | **Cost:** $10.00 per 1,000 searches + Claude model token usage fees. |
| `perform_crispr_cas9_genome` | **Bio Engineering** | Simulates the CRISPR-Cas9 process, including designing guide RNAs (gRNAs) and predicting editing outcomes for genomic loci. | Essential for gene editing experiment planning. |
| `query_alphafold` | **Database** | Retrieves predicted 3D protein structures from the AlphaFold Database using UniProt accession IDs. | Provides structures for proteins lacking experimental data. |
| `run_diffdock_with_smiles` | **Pharmacology** | Performs molecular docking of SMILES-format ligands to proteins using DiffDock. | Utilizes GPU acceleration for drug binding prediction. |
| `annotate_celltype_scRNA` | **Genomics** | Annotates cell types in scRNA-seq experiments using Leiden clustering and LLM-based reasoning. | Automates interpretation of single-cell transcriptomics. |
| `perform_flux_balance_analysis` | **Systems Biology** | Runs Flux Balance Analysis (FBA) on metabolic networks to predict growth and product yields. | Standard for metabolic engineering and systems biology. |
| `blast_sequence` | **Database** | Identifies DNA or protein sequences using BLAST against the NCBI database. | Fundamental for sequence alignment and homology searches. |

### 2.4.1 Biomni-R0-32B Model

Biomni-R0-32B is based on the Qwen3-32B architecture and has been fine-tuned for biomedical reasoning tasks. The model was trained using the Biomni-SFT dataset, a proprietary collection of biomedical instruction-response pairs that is not publicly available. Key characteristics include:

- **Base Architecture**: Qwen3-32B (32 billion parameters)
- **Training Data**: Biomni-SFT dataset (proprietary)
- **Optimization**: Fine-tuned for agentic reasoning patterns
- **Precision**: Original model distributed in BF16 format

It is important to note that while the original Biomni-R0-32B paper reports an overall accuracy of 66.9% on the Eval1 benchmark [4], our implementation targets the publicly available version provided via the official GitHub and HuggingFace repositories. Because the Biomni-SFT training data is proprietary and the full agent orchestration recipe used in the original study remains partially closed-source, local reproduction results may vary. Furthermore, our configuration reflects a resource-efficient version of the agent, utilizing only open-source tools and free APIs. This is the main challenge for this thesis. One does not need the training data to extract semantic knowledge from the fine-tuned model. The Biomni project also developed an agent system called Biomni-A1, which uses LangGraph [13] for orchestrating multi-step biomedical reasoning workflows. LangGraph is an abstraction layered on top of LangChain that allows us to author stateful, cyclic agent workflows using a graph-based abstraction. It is one of the most reliable libraries to build production-ready AI agents. Since our experiments utilize the base R0 model with the agentic recipe, architecture workflow provided by the authors of Biomni Agent A1, it is important to understand the context within which the model operates. This context is examined more thoroughly in the following subsection.

### 2.4.2 Eval1 Benchmark

The Eval1 benchmark is a thorough biomedical assessment that was curated by a group of biomedical scientists [14]. It was created as a benchmark for the Biomni R0 Model. There are 433 questions total, divided into ten different task categories:

1. **CRISPR Delivery**: Focuses on the mechanisms and optimization strategies for CRISPR-Cas9 delivery. (10 questions)
2. **GWAS Causal Gene - GWAS Catalog**: Utilizes the GWAS Catalog database to pinpoint causal genes within genome-wide association studies. (50 questions)
3. **GWAS Causal Gene - OpenTargets**: Employs the OpenTargets platform for the systematic identification of causal genes. (50 questions)
4. **GWAS Causal Gene - PharmaProjects**: Leverages pharmaceutical databases to extract drug targets from GWAS-derived data. (50 questions)
5. **GWAS Variant Prioritization**: prioritize/rank GWAS-associated variants by predicted functional relevance (e.g., regulatory effects, coding impact, gene proximity). (43 questions)
6. **Lab Bench DBQA**: answer natural-language questions over laboratory experiment databases (samples, protocols, results, metadata). (50 questions)
7. **Lab Bench SeqQA**: answer questions that require sequence-based analyses (e.g., similarity searches, motif finding, variant annotation). (50 questions)
8. **Patient Gene Detection**: infer candidate disease genes from patient clinical information (phenotypes, family history, and available molecular data). (50 questions)
9. **Rare Disease Diagnosis**: support differential diagnosis for rare genetic disorders by combining phenotypic profiles with candidate genes/variants. (30 questions)

10. **Screen Gene Retrieval**: retrieve and rank genes implicated by functional screening datasets (e.g., CRISPR or RNAi screens). (50 questions)

The benchmark is designed to assess more than straightforward fact retrieval; it specifically targets multi-step reasoning in biomedical settings. These questions are framed as agentic tasks, requiring the model to navigate simulated tools and databases to reach a correct answer.

### 2.4.3 Agentic Workflow Architecture Biomni-A1

To tackle complex biomedical questions that call for outside tools, the model works inside a circular state graph setup. As shown in Figure 2.1, this process drives a repeating "Reason-Act" cycle. The flow kicks off at the `__start__` node and shifts to the `generate` phase, where the LLM writes a reasoning path and chooses the next move. If the model creates an `<execute>` tag, the state jumps to the `execute` node, which runs the code or tool and feeds results back to the context. This cycle repeats until the model produces a `<solution>` tag, moving finally to `__end__`. The dotted lines represent conditional edges in which if the condition is satisfied, the model moves to this node, otherwise it moves to the next node. In case of ending the loop between `generate` and `execute` nodes, the model searches for a `<solution>` tag to move to the `__end__` state in which it terminates the workflow to provide the final answer within the solution tag.

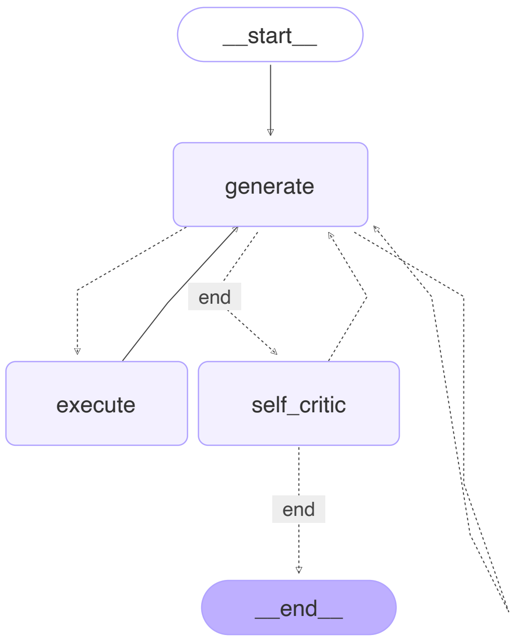

Figure 2.1: The Biomni-A1 agentic workflow implemented in LangGraph. The cycle between `generate` and `execute` allows for multi-step reasoning. The `self_critic` branch is disabled in our experiments to ensure computational efficiency.

### 2.4.4 Configuration of Self-Critic

While the design contains a `self_critic` node built to check and reject the model's own answers, we set this feature to **False** for all trials in this thesis. The self-correction cycle forces the model to read its whole thought path again and give feedback, which could spark many extra rounds of generation and execution. This repeating process makes the run time and token use spike significantly. Since the main goal of this work is **resource-efficient deployment** on limited, shared machines, running such heavy loops is not an option. For this reason, the agent leans on the model's built-in logic to focus on speed and energy efficiency.

## 2.5 The Orthogonality Hypothesis

The thesis' main theoretic point is the *Orthogonality Hypothesis*, which asserts: Quantization noise is high rank and isotropic, and semantic knowledge acquired by the extracted LoRA is low rank and structured. The two elements are mutually orthogonal in the weight space, which enables them to work together without major interference.

### 2.5.1 Theoretical Justification

Consider a weight matrix $W$ that undergoes both quantization and semantic adaptation:

$$W_{final} = W_0 + \Delta W_{semantic} + N_{quant} \tag{2.10}$$

where:

- $W_0$ is the original pre-trained weight
- $\Delta W_{semantic} = BA$ is the low-rank semantic adaptation (rank $r$)
- $N_{quant}$ is the quantization noise

The quantization noise $N_{quant}$ arises from rounding errors during quantization:

$$N_{quant} = W_{quantized} - W_{original} \tag{2.11}$$

This noise has several important properties:

1. **High-rank**: Quantization errors affect all elements of the weight matrix, resulting in a noise matrix with rank close to $\min(d, k)$.
2. **Isotropic**: The rounding errors are approximately uniformly distributed and uncorrelated with the semantic structure of the weights.
3. **Small magnitude**: For well-calibrated quantization, $||N_{quant}||_F \ll ||W_0||_F$.

The low-rank semantic adaptation $\Delta W_{semantic}$ has complementary properties:

1. **Low-rank**: By construction, $\text{rank}(\Delta W_{semantic}) = r \ll \min(d, k)$.
2. **Structured**: The adaptation captures specific semantic patterns (e.g., biomedical reasoning).
3. **Moderate magnitude**: Typically $||\Delta W_{semantic}||_F \approx 0.01 \times ||W_0||_F$.

### 2.5.2 Orthogonality Argument

If $N_{quant}$ and $\Delta W_{semantic}$ are orthogonal, their inner product should be small:

$$\langle N_{quant}, \Delta W_{semantic} \rangle = \text{Tr}(N_{quant}^T \Delta W_{semantic}) \approx 0 \tag{2.12}$$

This orthogonality implies that the two components do not interfere with each other. Specifically:

$$||W_0 + \Delta W_{semantic} + N_{quant}||^2 \approx ||W_0||^2 + ||\Delta W_{semantic}||^2 + ||N_{quant}||^2 \tag{2.13}$$

In practical terms, this means that a LoRA adapter trained on a BF16 model should remain effective when applied to the FP8-quantized version of the same base model, because the quantization noise does not corrupt the low-rank semantic subspace.

### 2.5.3 Theoretical Formalization of Subspace Orthogonality

To explain the robustness of the Naive Transfer method, we formalize the interaction between the semantic adaptation and quantization noise using high-dimensional geometry and linear algebra.

#### Definitions in Weight Space

Let the weight space be $\mathbb{R}^{d_{out} \times d_{in}}$. We define three matrices:

- **Base Weights** ($W_{base}$): The original pre-trained parameters.
- **Semantic Signal** ($L$): The LoRA update, where $L = BA$ and $\text{rank}(L) = r \ll \min(d_{out}, d_{in})$.
- **Quantization Noise** ($N$): The perturbation introduced by FP8 quantization, defined as $N = W_{FP8} - W_{BF16}$.

The final effective weight matrix during inference is:

$$W_{final} = W_{base} + L + N \tag{2.14}$$

#### The Orthogonality Condition

Our empirical findings indicate that the Frobenius inner product between the signal $L$ and noise $N$ is negligible:

$$\langle L, N \rangle_F = \text{Tr}(L^T N) \approx 0 \tag{2.15}$$

This geometric orthogonality implies that the quantization noise does not project significantly onto the semantic subspace spanned by the LoRA adapters.

#### Impact on Forward Pass

Consider an input activation vector $x \in \mathbb{R}^{d_{in}}$. The output activation $y$ is:

$$y = (W_{base} + L + N)x = \underbrace{W_{base}x}_{\text{Base Output}} + \underbrace{Lx}_{\text{Semantic Update}} + \underbrace{Nx}_{\text{Quantization Error}} \tag{2.16}$$

For the model to perform correctly, the Semantic Update must not be overwhelmed by the Quantization Error.

We can decompose the noise matrix $N$ into two components relative to the semantic subspace of $L$:

$$N = N_{||} + N_{\perp} \tag{2.17}$$

where:

- $N_{||}$: The component of noise parallel to $L$ (destructive interference)
- $N_{\perp}$: The component of noise orthogonal to $L$ (benign noise)

Since our cosine similarity is $\approx 0$, it implies that $||N_{||}||_F \approx 0$. Therefore, almost all quantization energy lies in $N_{\perp}$.

### Preservation of Semantic Direction

In high-dimensional spaces ($d \approx 5120$), random noise vectors are isotropic. According to the properties of high-dimensional geometry, a random vector (Noise) is nearly orthogonal to any fixed low-rank subspace (LoRA) with high probability.

We can analyze the Signal-to-Noise Ratio (SNR) within the specific semantic direction. Let $u$ be a singular vector of $L$ (a direction of semantic meaning). The effective SNR in this direction is:

$$\text{SNR}_{semantic} = \frac{||Lu||}{||P_L(N)u||} \tag{2.18}$$

where $P_L$ is the projection operator onto the subspace of $L$.

Because $N$ is high-rank (spread across all dimensions) and $L$ is low-rank ($r = 256$), the energy of $N$ is diluted across $d$ dimensions, while the energy of $L$ is concentrated in $r$ dimensions.

Even if the total magnitude of noise is large ($||N||_F > ||L||_F$), the projection of that noise onto the specific semantic direction is minimal:

$$||P_L(N)|| \approx \sqrt{\frac{r}{d}}||N|| \tag{2.19}$$

For Qwen-32B ($d = 5120$, $r = 256$):

$$\sqrt{\frac{256}{5120}} \approx 0.22 \tag{2.20}$$

This mathematical bound explains why Method A succeeds: The noise vector $N$ may be large, but it is "pointing" in directions that represent unused capacity in the weight space, rather than the specific directions required for biomedical reasoning.

## 2.5.4 Implications for Transfer Methods

The Orthogonality Hypothesis has direct implications for the three transfer methods investigated in this thesis:

- **Naive Transfer (Method A)**: Should succeed because $\Delta W_{semantic}$ is orthogonal to $N_{quant}$.
- **Corrective Extraction (Method B)**: May fail because attempting to encode both $\Delta W_{semantic}$ and $-N_{quant}$ in a rank-$r$ adapter exceeds the adapter's capacity, since $N_{quant}$ is high-rank.
- **Direct Quantization (Method C)**: Should largely succeed because quantization is applied after adaptation, preserving most of the semantic structure with some degradation from quantization.

The empirical validation of this hypothesis is a central contribution of this thesis, as presented in Chapters 5 and 6.

## 2.6 Related Work

This research is situated at the intersection of efficient Large Language Model (LLM) deployment, Post-Training Quantization (PTQ), and Parameter-Efficient Fine-Tuning (PEFT). While these components are well-studied individually, the interoperability between high-precision adapters and pre-quantized base models remains under-explored.

### 2.6.1 Efficient Deployment and FP8 Quantization

To cut the heavy running costs of LLMs, Post-Training Quantization (PTQ) and Quantization-Aware Training (QAT) has taken over as the go-to compression methods. Older tools like LLM.int8() [15] and SmoothQuant [10] aimed at managing activation spikes in integer files. Yet the field has moved to the FP8 floating-point type (E4M3/E5M2) [2], which runs natively on NVIDIA Hopper and Ada Lovelace chips. Block-wise quantization is widely used to reduce accuracy losses in very large models by applying scale factors to small weight blocks (e.g., 128 elements). While this typically keeps quantization error low, the resulting dense pattern of per-block scales complicates the weight-space arithmetic required to merge or combine adapters.

### 2.6.2 PEFT and the Training-Inference Gap

LoRA [3] builds on the idea of intrinsic dimensionality [11], which argues that adapting a model to a new task can often be achieved within a low-rank subspace. More recent methods that combine quantization with adaptation—such as QLoRA [16] and LoftQ [17]—primarily focus on making training more efficient. QLoRA trains LoRA adapters directly on top of a quantized base model, while LoftQ initializes the adapters so they compensate for quantization error before training begins.

*How this work differs:* In contrast to QLoRA and LoftQ, which require retraining and typically access to training data, *our work addresses a transfer setting*: we study how to use already-trained, high-precision adapters with a quantized base model without any additional training. Our *"Corrective Extraction" approach (Method B)* indicates that a post hoc attempt to mimic LoftQ-style error compensation with a fixed-rank adapter runs into a capacity bottleneck. In effect, the quantization error behaves like a *high-rank disturbance* that overwhelms the adapter's *low-rank structure*, so the adapter cannot absorb the noise without exceeding its effective capacity.

### 2.6.3 Orthogonality and Biomedical Adaptation

Prior work on activation quantization has largely focused on preserving outlier dimensions. In contrast, we introduce an *Orthogonality Hypothesis* grounded in high-dimensional geometry. Specifically, we posit that isotropic quantization noise is, in a statistical sense, orthogonal to the structured, low-rank parameter shifts associated with semantic adaptation. To test the hypothesis in a demanding setting, we evaluate on biomedical tasks using the Biomni benchmark [14]. Tasks such as CRISPR design/optimization and GWAS interpretation often involve multi-step reasoning and have high consequences, so they provide a strict test of robustness. Biomedical workloads can also be sensitive to quantization error, meaning that small losses in numerical precision can lead to measurable drops in performance.

# Chapter 3

# Methodology

## 3.1 Experimental Design Overview

This chapter describes the setup used to study how LoRA updates can be transferred when the base model and the adapter are at different precision levels. We test three methods: Naive Transfer (Method A), Corrective Extraction (Method B), and Direct Quantization (Method C). Each method tests a different pathway by which fine tuning information may be preserved or degraded when different parts of the model use different numerical formats. We run the experiments with two separate implementations. The first uses vLLM, a standard inference system that applies optimized kernels and manages low level details automatically, and it is used mainly for Method A. The second approach uses a custom pipeline that reconstructs and modifies quantized weights directly. This allows access to internal model representations and is required for Method B. We describe each method in the same format, with the same set of implementation and execution details:

1. **Mathematical Formulation**: theoretical motivation and the tensor operations used by the method.
2. **Hypothesis**: the expected relationship between quantization noise and the semantic signal in the adapter.
3. **Implementation**: the procedure and engineering details used to run the method, including pseudocode for the core logic.

Our experimental setup comprises the following components:

- **Base model**: Qwen3-32B [1], a 32B-parameter state-of-the-art language model.
- **Domain-adapted model**: Biomni-R0-32B [4], a biomedical reasoning model derived from Qwen3-32B.
- **Quantization target**: FP8 (E4M3) [2] with block-wise quantization (Block-128).
- **Adapter type**: LoRA [3] with rank 256.
- **Evaluation domain**: biomedical reasoning tasks from the Eval1 benchmark.

Baseline performance is defined by Biomni-R0-32B evaluated in full BF16 precision, which serves as the gold-standard reference point for all subsequent comparisons on the Eval1 benchmark.

## 3.2 Baseline: Biomni-R0-32B (BF16)

Biomni-R0-32B serves as our gold standard reference model [4]. It is a domain-adapted version of Qwen3-32B specifically fine-tuned for biomedical reasoning tasks by the Biomni Project at Stanford University [12]. The model demonstrates superior performance on:

- CRISPR delivery mechanism questions
- Genetic variant interpretation and prioritization
- Genome-wide association studies (GWAS) causal gene identification
- Laboratory biology database question answering
- Sequence analysis and interpretation
- Patient gene detection from clinical data
- Rare disease diagnosis
- Functional screening gene retrieval

The model was trained using the Biomni-SFT dataset, a proprietary collection of biomedical instruction-response pairs. While we do not have access to the training data, the model weights are publicly available on HuggingFace [4]. The model is stored in BF16 precision with a total size of approximately 64 GB. All experimental methods aim to achieve comparable performance while reducing the memory footprint through FP8 quantization.

The baseline performance of 44.5% established in this study represents the model's "offline" reasoning and local tool-use capability. This deviates from the 66.9% reported in the original paper primarily due to the exclusion of the `advanced_web_search_claude` tool [4]. The original study relied heavily on this paid tool for real-time information retrieval; by excluding it, we test the intrinsic knowledge of the model and its ability to navigate free, open-source biomedical databases. This creates a more challenging environment for the agent but ensures the deployment remains cost-effective and resource-efficient. Also, the source code for evaluating the model is not publically available. The authors of the original study provided only a high level description of the evaluation pipeline.

## 3.3 Method A: Naive Transfer

### 3.3.1 Conceptual Approach

Method A, "Naive Transfer," assumes that quantization noise and semantic adaptation occupy different directions in weight space. The method computes a LoRA adapter from the BF16 weight difference between the domain adapted model and the base model, and then applies this adapter to the FP8 quantized base model.

### 3.3.2 Mathematical Formulation

The LoRA extraction is performed as follows:

$$L_{bio} = W_{biomni}^{BF16} - W_{qwen}^{BF16} \tag{3.1}$$

where $W_{biomni}^{BF16}$ represents the weights of Biomni-R0-32B and $W_{qwen}^{BF16}$ represents the weights of the base Qwen3-32B model, both in BF16 precision.

The extracted difference $L_{bio}$ is then approximated using low-rank decomposition:

$$L_{bio} \approx B \cdot A \tag{3.2}$$

where $B \in \mathbb{R}^{d \times r}$ and $A \in \mathbb{R}^{r \times k}$ with rank $r = 256$.

Finally, the LoRA is applied to the FP8-quantized base model:

$$W_{final} = W_{qwen}^{FP8} + B \cdot A \tag{3.3}$$

Note that this creates a mixed-precision model where the base weights are in FP8 and the LoRA adapter is in BF16.

### 3.3.3 Hypothesis

The key hypothesis underlying Method A is that:

$$W_{biomni}^{BF16} - W_{qwen}^{BF16} \approx W_{biomni}^{FP8} - W_{qwen}^{FP8} \tag{3.4}$$

This equivalence holds if the quantization noise $N_{quant}$ is approximately the same for both models and cancels out in the subtraction:

$$(W_{biomni}^{BF16} + N_{biomni}) - (W_{qwen}^{BF16} + N_{qwen}) \approx W_{biomni}^{BF16} - W_{qwen}^{BF16} \tag{3.5}$$

when $N_{biomni} \approx N_{qwen}$.

### 3.3.4 Implementation

Method A is implemented using MergeKit's [18] LoRA extraction functionality:

1. Load Qwen3-32B-BF16 and Biomni-R0-32B-BF16
2. Compute weight differences layer-by-layer
3. Apply SVD to extract rank-256 approximation using MergeKit toolkit [18] and Sanitize the adapter to remove embedding and normalization parameters.
4. Save as LoRA adapter in SafeTensors format
5. Apply adapter to Qwen3-32B-FP8 using vLLM [5]

Both the MergeKit toolkit's `extract_lora` [18] method and the sanitization pipeline will be discussed more thoroughly in Method B.

## 3.4 Method B: Corrective Extraction

### 3.4.1 Conceptual Approach

Method B, defined as "Corrective Extraction", seeks to develop a LoRA adapter that encodes domain-specific knowledge while also correcting quantization mistakes. The method derives a LoRA from the difference between the BF16 domain-adapted model and the dequantized FP8 base model.

### 3.4.2 Mathematical Formulation

The corrective LoRA is extracted as:

$$L_{corrective} = W_{biomni}^{BF16} - \text{Dequant}(W_{qwen}^{FP8}) \tag{3.6}$$

where Dequant$(\cdot)$ converts FP8 weights back to BF16 by applying the inverse quantization:

$$\text{Dequant}(W^{FP8}) = W^{FP8} \times s_{expanded} \tag{3.7}$$

The extracted difference includes both the semantic adaptation and the quantization error:

$$L_{corrective} = (W_{biomni}^{BF16} - W_{qwen}^{BF16}) + (W_{qwen}^{BF16} - \text{Dequant}(W_{qwen}^{FP8})) \tag{3.8}$$

$$L_{corrective} = \Delta W_{semantic} - N_{quant} \tag{3.9}$$

This combined difference is then approximated using rank-256 decomposition and applied to the FP8 base model:

$$W_{final} = W_{qwen}^{FP8} + B \cdot A \tag{3.10}$$

### 3.4.3 Hypothesis

Method B is motivated by the hypothesis that a rank-256 adapter has sufficient capacity to encode two distinct components at once: the low-rank semantic adaptation $\Delta W_{semantic}$ and an additional correction term that counteracts quantization distortion, $-N_{quant}$. Formally, we test whether a single LoRA update can represent both of the following:

1. The semantic adaptation $\Delta W_{semantic}$ (low-rank)
2. The quantization correction $-N_{quant}$ (high-rank)

If this assumption holds, the resulting model should remain domain-adapted while also compensating for a substantial fraction of the quantization-induced distortion, thereby behaving closer to a dequantized model at inference time.

### 3.4.4 Implementation

Method B differs from Method A in that it works on the quantized weight tensors, since the computations are performed in weight space. We built a dequantization pipeline and separated the workflow into three stages.

#### Step 1: Tensor Reconstruction (The Bridge Model)

Frameworks such as Transformers and vLLM aim for high throughput inference and automate most operations. They are built for speed and efficiency, so they hide the internal structure and storage details of compressed model weights, providing access only through standard prediction functions. This design works well for inference, but it complicates Corrective Extraction, which requires direct access to and modification of the model weights. Specifically, computing the corrective LoRA requires an element-wise difference between the adapted BF16 model and the FP8 base model:

$$\Delta W = W_{adapted}^{(BF16)} - W_{base}^{(FP8)}. \tag{3.11}$$

Existing tools and libraries generally lack a simple method to access the fundamental FP8 data and their corresponding block-specific scaling parameters in a way that allows for direct mathematical manipulation. The Qwen-32B-FP8 model is distributed as SafeTensors files in which each layer stores both the quantized weights and the corresponding scaling tensors:

- Weight tensors with dtype `float8_e4m3fn` (FP8 E4M3 format)
- Scale tensors with dtype `float32` and suffix `_input_scale_inv`

The naming convention `_input_scale_inv` indicates that the stored values are *inverse scales*, which implies the reconstruction rule:

$$W_{BF16} = W_{FP8} \times s_{inv}. \tag{3.12}$$

To recover the correct quantization layout, we systematically inspected tensor shapes across layers and identified the Block-128 scheme:

| Layer | Weight Shape | Scale Shape |
|---|---|---|
| `model.layers.0.mlp.gate_proj` | $[5120, 13696]$ | $[40, 107]$ |
| `model.layers.0.mlp.up_proj` | $[5120, 13696]$ | $[40, 107]$ |
| `model.layers.0.self_attn.q_proj` | $[5120, 5120]$ | $[40, 40]$ |

Table 3.1: Weight and scale tensor shapes revealing Block-128 quantization

The ratios $\frac{5120}{40} = 128$ and $\frac{13696}{107} \approx 128$ confirm the block size. We implemented the following dequantization algorithm:

```python
def expand_scale(scale, target_shape):
    """Expand scale tensor to match target weight shape."""
    block_H = target_shape[0] // scale.shape[0]
    block_W = target_shape[1] // scale.shape[1]
    expanded = scale.repeat_interleave(block_H, dim=0)
    expanded = expanded.repeat_interleave(block_W, dim=1)
    if expanded.shape != target_shape:
        expanded = expanded[:target_shape[0], :target_shape[1]]
    return expanded

def dequantize_layer(weight_fp8, scale_inv):
    """Dequantize a single layer."""
    scale_expanded = expand_scale(scale_inv, weight_fp8.shape)
    weight_float = weight_fp8.to(torch.float32)
    weight_dequant = weight_float * scale_expanded
    return weight_dequant.to(torch.bfloat16)
```

Listing 3.1: Block Dequantization Algorithm

### Step 2: Statistical Verification (The Autopsy Protocol)

To avoid expensive GPU runs during early checks, we validated the reconstructed tensors on CPU to catch corruption. In a correctly reconstructed layer, weights are typically centered near zero with a moderate variance. When reconstruction fails, the mean shifts away from zero, values clip or saturate, or the variance becomes unusually large. These simple statistics helped identify issues such as incorrect scaling factors, reshape errors, and dtype conversion mistakes.

- Mean: $\mu \approx 0.0$ (typically $|\mu| < 0.01$)
- Standard Deviation: $\sigma \approx 0.02$ to $0.05$
- Min/Max: Within $[-0.5, 0.5]$ for most layers

```python
def autopsy_layer(weight, layer_name):
    """Perform statistical autopsy on a weight tensor."""
    stats = {
        'mean': weight.mean().item(),
        'std': weight.std().item(),
        'min': weight.min().item(),
        'max': weight.max().item(),
    }
    if abs(stats['mean']) > 1.0:
        print(f"WARNING: {layer_name} has exploded mean")
    if stats['std'] > 10.0:
        print(f"WARNING: {layer_name} has exploded std")
    return stats
```

Listing 3.2: Weight Autopsy Verification

This iterative procedure enabled rapid debugging and converged on a correct reconstruction pipeline:

1. **Iteration 1**: Per-tensor scaling $\rightarrow$ weight magnitudes exploded (Mean $> 448$)
2. **Iteration 2**: Incorrect block size (64) $\rightarrow$ shape mismatch during reconstruction
3. **Iteration 3**: Correct block size (128) but incorrect scale application $\rightarrow$ systematically biased weights
4. **Iteration 4**: Correct implementation $\rightarrow$ Healthy statistics (Mean $\approx 0$, Std $\approx 0.02$)

### Step 3: LoRA Extraction and Sanitization

Once the bridge model was converted to a single BF16 format, we isolated what the biomedical training added. Using MergeKit [18] extract_lora method, we extracted a low-rank adapter (LoRA) that approximates the difference between the biomedical model weights and the reconstructed bridge weights. This is written as:

$$\Delta W = W_{biomni}^{BF16} - W_{bridge}^{BF16} \approx U_{256}\Sigma_{256}V_{256}^{T} = B \cdot A. \tag{3.13}$$

In this formulation, $\Delta W$ is approximated with a rank-256 factorization, which gives LoRA matrices $A$ and $B$ that represent the transferred update.

This adapter could not be used as-is. When we tried to load it through vLLM's adapter interface, it failed at runtime because vLLM's FP8 kernels restrict which modules can receive LoRA updates [19]. Historically, vLLM's LoRA support has omitted certain transformer components—including `lm_head`, `embed_tokens`, normalization layers, and bias terms—because these modules are treated differently from the attention and feed-forward blocks that LoRA typically targets. One limitation is that `embed_tokens` and `lm_head` act on full vocabulary sized tensors rather than hidden state sized tensors. This often calls for a different low rank parameterization, for example separate LoRA matrices designed for embedding type weights. Early vLLM LoRA support did not implement this special case, which could cause adapter load failures, including missing tensors and shape mismatches. In practice, embedding-related logic can also interact with extended-vocabulary checkpoints (for example, separate files like `new_embeddings.safetensors`); although vLLM identifies embedding modules (e.g., via an `EMBEDDING_MODULES` check), it may reject adapters when the expected embedding LoRA weights are incomplete. Finally, normalization and bias parameters are commonly absent from standard LoRA configurations or do not map cleanly to rank-based adapters, and vLLM prioritizes adapting attention and MLP/FFN weights for efficiency.

We therefore implemented a *sanitization pipeline* that removes LoRA parameters attached to unsupported modules:

- **Load and inspect**: the LoRA state dictionary was loaded into CPU memory to inspect parameter keys.
- **Define exclusions**: a forbidden-substring list was specified as `["lm_head", "embed_tokens", "norm", "bias"]`.
- **Prune incompatible tensors**: any tensor key containing one of the forbidden substrings was removed.
- **Update adapter configuration**: `adapter_config.json` was edited to set `modules_to_save = null`.

This sanitization step introduces a necessary trade-off. Information learned in the embedding layers, normalization parameters, or bias terms is removed. This means the adapter keeps only part of the residual update, not the full change.

This loss of information likely degrades transfer fidelity and is a probable contributor to the higher error rate observed for Method B. Despite the fact that the sanitization step was also performed in Method A, the performance degradation is not as severe as in Method B. This is because in Method A, the adapter is applied on top of the base model in BF16 precision, which is more faithful to the original weights compared to the quantized weights in Method B.

## 3.5 Method C: Direct Quantization

### 3.5.1 Conceptual Approach

Method C, termed "Direct Quantization," takes the most direct route to low-precision deployment. Rather than attaching an adapter to a quantized base model, we quantize the already domain-adapted Biomni-R0-32B model itself. In practice, this means applying post-training quantization to convert the adapted BF16 weights directly into FP8. Because quantization can distort the specialized behavior learned during adaptation, we calibrate the procedure on biomedical-domain data, aiming to choose FP8 scaling factors that are tuned to the activation patterns the model exhibits on biomedical reasoning traces.

### 3.5.2 Mathematical Formulation

The quantization procedure is defined as:

$$W_{biomni}^{FP8} = \text{Quant}(W_{biomni}^{BF16}, \mathcal{D}_{calibration}), \tag{3.14}$$

where $\mathcal{D}_{calibration}$ denotes a biomedical text dataset used to estimate the quantization scales.

### 3.5.3 Hypothesis

Method C is motivated by the hypothesis that calibration on biomedical-domain data produces quantization scales that are better matched to the activation statistics of biomedical reasoning than scales derived from generic text. If this holds, then directly quantizing the adapted model should retain more of its domain-specific performance than an FP8 model calibrated out of domain, while still achieving a substantially smaller memory footprint.

### 3.5.4 Implementation

We implemented Method C using the `llm-compressor` library [20] (formerly the vLLM compressor) on an NVIDIA H100 GPU. The goal was to mirror the FP8 deployment format used by Qwen while keeping the adapted model as faithful as possible after quantization. To that end, we applied fine-grained FP8 quantization in the E4M3 format with Block-128 sub-blocking, which estimates scale and bias parameters locally rather than at the level of an entire tensor, thereby better preserving weight statistics [2, 10].

The resulting configuration can be summarized as follows:

- **Quantization Format**: We quantized weights to FP8 E4M3 using a fine-grained blocked scheme.
- **Block Size**: We used a block size of 128 to match the official Qwen3-FP8 tensor format and ensure compatibility with existing tooling.
- **Calibration Protocol**: Instead of calibrating on short context windows (e.g., 512 or 1024 tokens), we deliberately prioritized full-trajectory calibration in order to capture long-context activation outliers that occur in biomedical multi-step reasoning.
  - **Sampling Strategy**: We performed stratified random sampling across all 10 Eval1 tasks to maintain balanced coverage (approximately 12–13 samples per task), restricted to successful trajectories.
  - **Volume**: This produced a calibration set of 123 full-context instances totaling 3,163,274 tokens.
  - **Context Depth**: The mean sample length was 25,718 tokens (77,603 characters), and the longest contexts reached 75,508 tokens. This “deep calibration” was intended to yield FP8 scaling factors that remain stable under the activation patterns induced by long, structured reasoning traces, substantially exceeding typical calibration practice.
- **Excluded Layers**: We excluded `lm_head` from FP8 quantization and retained it in higher precision for stability.

The quantization recipe used was:

```python
# Configuring FP8 Recipe (Block Size 128)
recipe = QuantizationModifier(
    targets="Linear",
    scheme="FP8",
    ignore=["lm_head"]
)
```

This setup uses fine grained FP8 quantization with Block 128 granularity for all linear layers except the language model head. Each weight matrix is partitioned into contiguous blocks of 128 elements along the quantization dimension, and a separate scale, and a bias when applicable, is computed for each block. Because the quantization parameters are estimated per block rather than per tensor, the procedure follows local changes in weight statistics across the matrix more closely than per tensor quantization. The format matches the official Qwen3 32B FP8 release, which ensures compatibility and improves accuracy. For comparison, we also evaluated an INT8 quantized version of R0 32B from HuggingFace. That model uses standard per tensor INT8 quantization [21], a common `llm-compressor` baseline, and provides a reference point for assessing the effect of Block 128 FP8 quantization.

## 3.6 Evaluation Framework

### 3.6.1 Eval1 Benchmark

We evaluate all methods on Eval1, a biomedical reasoning benchmark from the Biomni Project [14]. Eval1 has 433 questions in 10 task categories and covers a range of biomedical settings and reasoning demands. It is not designed mainly for factual recall. Many questions require multiple reasoning steps, and some involve simulated tools or database style resources before the final answer is produced.

### 3.6.2 Evaluation Metrics

We report the following metrics:

- **Accuracy**: Percentage of correct answers across all 433 questions
- **Retention Percentage**: $\frac{\text{Accuracy}_{method}}{\text{Accuracy}_{baseline}} \times 100\%$
- **Per-Task Breakdown**: Performance on each of the 10 task categories
- **Memory Footprint**: Model size in GB

### 3.6.3 Inference Configuration

All models are evaluated using vLLM [5] with the following configuration:

- **Temperature**: 0.6 (temperature sampling)
- **Max Tokens**: 32,768 (to accommodate full reasoning trajectories)
- **Max Model Length**: 65,536 (2x the 32K base)
- **Max Number of Batched Tokens**: 32,768 (half of max for batching efficiency)
- **Top-p**: 0.95
- **Top-k**: 20
- **GPU**: 4× NVIDIA A100 80GB / H100 94GB

### 3.6.4 Multi-GPU Deployment with Load Balancing

To run the 433 Eval1 questions in parallel, we deployed four separate vLLM server instances, each assigned to a different GPU. Each server was started with LoRA support enabled:

```
CUDA_DEVICES=0 python -m vllm.entrypoints.openai.api_server \
    --model $MODEL_PATH \
    --port 8000 \
    --dtype bfloat16 \
    --max-model-len 65536 \
    --enable-lora \
    --max-lora-rank 256 \
    --lora-modules $LORA_NAME=$LORA_PATH \
    --gpu-memory-utilization 0.90 \
    --rope-scaling '{"rope_type":"yarn","factor":2.0, \
        "original_max_position_embeddings":32768}' \
    --max-num-batched-tokens 32768
```

Listing 3.3: vLLM Server Launch (per GPU)

A custom load balancer distributed requests across four backends on ports 8000 to 8003. It monitored the number of in flight requests per backend and routed each new request to the backend with the shortest queue, which avoided overloading a single GPU:

```
def get_least_loaded_backend():
    """Select backend with fewest in-flight requests."""
    with state_lock:
        healthy = [b for b in BACKENDS
                   if backend_state[b]["status"] == "healthy"]
        candidates = healthy if healthy else BACKENDS
        min_backend = min(candidates,
                    key=lambda b: backend_state[b]["in_flight"])
        backend_state[min_backend]["in_flight"] += 1
        return min_backend
```

Listing 3.4: Least-Loaded Backend Selection

This setup allowed concurrent evaluation and failover. If a backend stopped responding, the load balancer sent new requests to the remaining healthy instances.

### 3.6.5 Client-Side Parallel Request Dispatch

On the client side, we used a multiprocess asynchronous evaluation framework to maximize throughput. It spawns 18 worker processes, and each worker maintains 4 concurrent requests using Python's `asyncio`, for a theoretical peak of 72 simultaneous agent runs:

```
NUM_PROCESSES = 18
ASYNC_CONCURRENCY_PER_PROCESS = 4

async def run_evaluation_for_process(process_id, instance_queue):
    evaluator = BiomniR0EvaluatorB1(process_id)
    semaphore = asyncio.Semaphore(ASYNC_CONCURRENCY_PER_PROCESS)

    async def process_with_semaphore(instance):
        async with semaphore:
            result = await evaluator.process_instance(instance)
            evaluator.save_result_and_update_stats(result)

    # Process instances from shared queue
    while not instance_queue.empty():
        instance = instance_queue.get_nowait()
        if evaluator.try_claim_instance(instance['instance_id']):
            asyncio.create_task(process_with_semaphore(instance))
```

Listing 3.5: Multi-Process Async Configuration

The design incorporates several reliability features:

- **Atomic instance claiming**: A file-lock mechanism prevents duplicate processing when multiple processes attempt to claim the same instance.
- **Incremental persistence**: Results are written to disk immediately after each completion, ensuring no data loss on crashes.
- **Resumable execution**: Previously processed instances are skipped on restart, enabling long evaluations to be paused and resumed.

## 3.7 Experimental Predictions

Based on the Orthogonality Hypothesis, we make the following predictions:

1. **Method A (Naive Transfer)**: Should achieve $> 95\%$ retention because quantization noise is orthogonal to semantic adaptation.
2. **Method B (Corrective Extraction)**: May fail ($< 80\%$ retention) because the rank-256 adapter lacks capacity to encode both semantic knowledge and high-rank quantization correction.
3. **Method C (Direct Quantization)**: Should achieve $> 90\%$ retention because quantization is applied after adaptation with domain-specific calibration, though some degradation is expected from the quantization process itself.

The results for these predictions are analyzed in Chapter 4. The discussion for the results and the hypothesis are in Chapter 5.

# Chapter 4

# Results

This chapter presents the empirical evaluation of the three cross-precision transfer methods described in Chapter 3. We report quantitative performance on the Eval1 biomedical benchmark, provide per-task breakdown analysis, and present visualizations of the results. The findings provide empirical support for the Orthogonality Hypothesis and reveal the limitations of corrective extraction approaches.

## 4.1 Evaluation Setup

### 4.1.1 Hardware and Software Configuration

All experiments were conducted on the following hardware:

- **GPU**: NVIDIA A100 80GB (SXM4) / NVIDIA H100 94GB
- **CPU**: AMD EPYC 7763 (64 cores)
- **RAM**: 512GB DDR4
- **Storage**: NVMe SSD (10TB)

Software versions:

- **vLLM** [5]: 0.5.4
- **PyTorch**: 2.3.0
- **CUDA**: 12.1
- **Transformers**: 4.41.0
- **MergeKit** [18]: 0.4.2
- **llm-compressor** [20]: 0.2.0

## 4.2 Robust Response Annotation Protocol

A major issue during the biomedical reasoning evaluation was inconsistent output formatting. Although the prompt required the final answer to be wrapped in `<solution>` tags, the models often failed to follow this requirement, especially after quantization or after applying an adapter. With strict string matching in the initial evaluation, many correct outputs were counted as false negatives. In many cases, models arrived at the correct biomedical conclusion but expressed it in free-form natural language (e.g., "Therefore, the gene **PYCARD** emerges as the optimal candidate...") rather than emitting the required XML answer format between the `<solution>` tags. To reduce false negatives from strict formatting rules, we used a four stage annotation pipeline.

### 4.2.1 Step 1: Heuristic Extraction

We designed an answer extraction module that checks the last 200 tokens of each response. It does not rely only on XML-style tags. Instead, it searches for phrases that usually mark the end of an answer, such as headings like “Conclusion” or “Final Answer”, and for closing statements (e.g., “X emerges as . . . ”) or simple copula patterns (“X is . . . ”, “X has . . . ”) followed by a candidate entity. In parallel, the module computes a ground-truth density score by counting how often the known correct answer string appears in the same segment. It also handles indirect mentions, where the model refers to the target using a synonym, alias, or related identifier, for example a SNP ID such as `rs4949874` instead of a gene symbol.

### 4.2.2 Step 2: LLM-as-a-Judge Evaluation

For cases where the heuristics produced low confidence, we used a separate large language model as an automatic judge (`openai-gpt-oss-120b`, accessed via the GWDG Chat AI platform [22]). For each uncertain example, we constructed an evaluation prompt that included the original question, the reference answer, the model’s full output, and the heuristic signals from Step 1. The prompt asked the judge to assess semantic equivalence rather than surface form, so that differences in formatting (e.g., XML, JSON, or plain text) did not matter, while factual correctness still did.

```
You are an expert evaluator for a biomedical automated reasoning task.
Your goal is to determine if the
**Model's Extracted Solution** matches the **Ground Truth Answer**.

### 1. Task Context (Original Question Snippet)
{original_prompt}

### 2. Ground Truth Answer
{answer}

### 3. Model's Extracted Solution
{extracted}

### 4. Heuristic Signals
- Answer Occurrences in last 200 tokens: {count}
- Potential Answer (from 'Conclusion'): {conclusion_candidate}
- Potential Answer (emerges): {emerges_candidate}

### Evaluation Rules
1. Ignore Formatting: If Truth is "CCR4" and Solution is "{'gene': 'CCR4'}",
this is a MATCH (1).
2. Multiple Choice: If Truth is "B", and Solution is "B. P335A", this is a MATCH (1).
3. Use Heuristics: If statistics show the answer appeared in the "Conclusion",
trust the model found the answer.
4. Nulls: Only return 'null' if the solution is completely ambiguous.

Output Format:
Reasoning: <Explanation>
Label: <1, 0, or null>
```

Listing 4.1: Prompt Template for LLM-based Evaluation Judge

If the judge could not make a confident determination, it was instructed to return `null`, thereby flagging the instance for manual review.

### 4.2.3 Step 3: Human Verification and Web Search

Although the automated judge substantially reduced False Negatives, it can still produce False Positives through semantic over-matching and it occasionally returns `null`. To protect data integrity, we therefore manually verified every flagged case. We used Perplexity AI to run targeted web searches for biological validation, which was particularly important when the model output was biologically correct but not string-identical to the ground truth (e.g., a gene synonym, a protein name rather than a gene name, or a nearby causal variant instead of the canonical target). During this step, we also checked that the model output actually *endorsed* the entity as the answer, rather than merely mentioning it incidentally or in a negated context (e.g., "Gene X is NOT the cause").

### 4.2.4 Step 4: Final Aggregation

All performance statistics reported in this thesis reflect the aggregated outcomes of this multi-stage pipeline. This evaluation procedure is intended to measure underlying reasoning performance rather than adherence to a rigid output schema.

## 4.3 Quantitative Performance Results

### 4.3.1 Overall Performance Comparison

Table 4.1 summarizes the primary quantitative comparison across all experimental methods and references.

| **Model Configuration** | **Method** | **Acc.** | **Ret.** | **Size** |
|---|---|---|---|---|
| Biomni-R0-32B (BF16) | Baseline | 44.5% | 100.0% | 64.0 GB |
| Qwen3-FP8 + LoRA 256 | Method A | **44.6%** | **100.2%** | 40.0 GB |
| Qwen3-FP8 + LoRA 256 (Dequant) | Method B | 29.5% | 66.3% | 40.0 GB |
| R0-FP8 (Block-128) | Method C | 41.4% | 93.0% | 32.0 GB |
| Qwen3-FP8 + LoRA 128 | Variant | 43.3% | 97.3% | 40.0 GB |
| R0-INT8 (Per-tensor) | Reference | 40.9% | 91.9% | 34.8 GB |
| Qwen3-BF16 (No Adapt.) | Reference | 22.1% | 49.7% | 64.0 GB |

Table 4.1: Performance comparison on Eval1 (433 questions). Method A achieves statistical parity with the baseline (44.6% vs. 44.5%) with ∼37.5% memory reduction. Acc. = Accuracy, Ret. = Retention.

The baseline accuracy in our evaluation is 44.5%, which differs from the 66.9% reported in the original Biomni study [4]. The main reason is tool availability. As shown in Table 2.1, we did not include `advanced_web_search_claude`, a tool that enables reasoning based web retrieval, because its cost is $10.00 per 1,000 searches. To keep the evaluation reproducible and resource efficient, we restricted the agent to open source tools and free API tiers. Under these constraints, 44.5% is the highest accuracy reached in a fully open source and local setup, and we use it as the baseline for the cross precision transfer experiments. There are additional limits on replication. Although the Biomni R0 32B model weights are available on HuggingFace [4], the Biomni SFT dataset used for training is proprietary. In addition, only parts of the original agent workflow and orchestration logic have been released. As a result, the absolute accuracy can vary with the evaluation setup, the set of available tools, and the configuration. The three methods show clear differences. **Method A Naive Transfer** reaches a mean accuracy of 44.6%, which matches the BF16 baseline of 44.5% within the noise of the evaluation, so the transfer does not cause a measurable loss even when a BF16 LoRA adapter is used with an FP8 base model. **Method B Corrective Extraction** performs much worse, with 29.5% accuracy and 66.3% retention, and it is below the result from direct quantization. This supports the view that a rank 256 adapter is not sufficient to both preserve the learned biomedical adaptation and correct the errors introduced by quantization, especially after sanitization removes part of the update. Sanitization was applied in both Method A and Method B to make the models compatible with vLLM inference [19]. vLLM LoRA support has historically omitted several transformer components, including `lm_head`, `embed_tokens`, normalization layers, and bias terms, which required pruning these parameters from extracted adapters. Even though both methods use the same pruning step, Method B is affected more because the quantization correction signal is spread across many weight matrices, including those that are pruned, while Method A mainly stores domain adaptation in attention and MLP weights that remain after pruning. **Method C Direct FP8 Quantization** falls between the other two methods. It reaches 41.4% accuracy and 93% retention using Block 128 FP8 quantization calibrated on biomedical data, but it is still 3.2 percentage points below Method A. This suggests that keeping a high precision adapter on top of a quantized base model is more effective than quantizing the entire specialized model. For deployment, Method A also reduces memory use. The total requirement drops from 64 GB to about 32.5 GB when LoRA overhead is included, which makes serving on a single GPU feasible with much lower VRAM. Results from the rank 128 variant point in the same direction. Accuracy remains strong at 43.3%, which corresponds to 97.3% retention, indicating that most domain specific knowledge can be retained without the full capacity of rank 256.

### 4.3.2 Performance Visualization

Figure 4.1 provides a visual summary of average performance across the evaluated configurations.

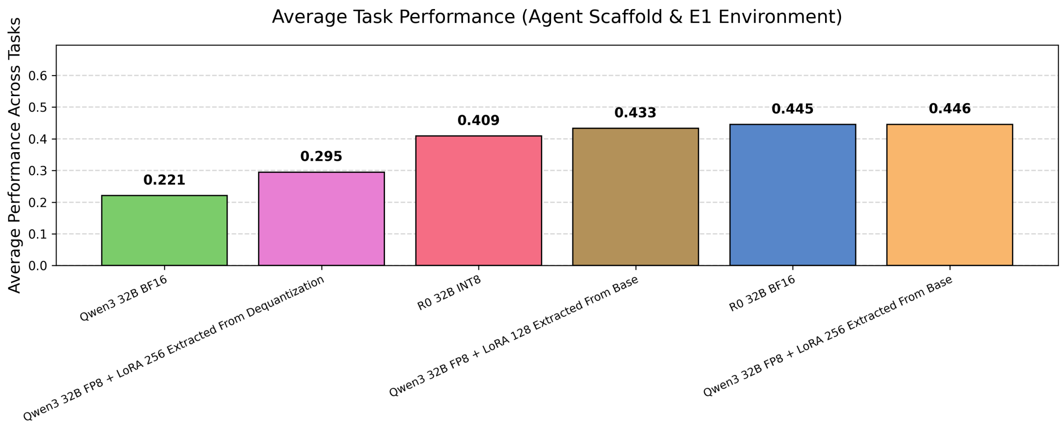

Figure 4.1: Average task performance across all model configurations. Method A (Qwen3 32B FP8 + LoRA 256 Extracted From Base) achieves 44.6% accuracy, closely matching the baseline R0 32B BF16 (44.5%).

### 4.3.3 Per-Task Performance Breakdown

Table 4.2 reports accuracy disaggregated by task category, and Figure 4.2 visualizes the same results as a heatmap to highlight systematic strengths and failures across configurations.

Key observations from the per-task analysis include:

- **Method A consistency**: Across most categories, Method A functions as a near drop-in replacement for the full-precision baseline. It matches or exceeds baseline performance on 8 out of 10 tasks, and when deviations occur they tend to be small (typically within 1–8 percentage points).
- **Method B systematic failure**: By comparison, Method B exhibits a broad and consistent decline across every task category, pointing to a general degradation of useful representations rather than a narrow, task-specific weakness. The most pronounced failures appear on reasoning-intensive tasks that demand longer multi-step inference, including Patient Gene Detection (12% vs. 32% baseline) and Rare Disease Diagnosis (10% vs. 37% baseline).

| **Task** | **Baseline (R0 BF16)** | **Method A (Qwen3 FP8+LoRA)** | **Method B (Dequant)** | **Method C (R0 INT8)** | **Base (Qwen3 BF16)** |
|---|---|---|---|---|---|
| CRISPR Delivery | 20% | 30% | 20% | 20% | 30% |
| GWAS Causal Gene (Catalog) | 36% | 36% | 18% | 42% | 14% |
| GWAS Causal Gene (OpenTargets) | 60% | 54% | 30% | 68% | 54% |
| GWAS Causal Gene (Pharma) | 50% | 56% | 48% | 56% | 50% |
| GWAS Variant Prioritization | 44% | 49% | 21% | 47% | 7% |
| Lab Bench DBQA | 58% | 54% | 46% | 52% | 10% |
| Lab Bench SeqQA | 74% | 66% | 60% | 62% | 24% |
| Patient Gene Detection | 32% | 28% | 12% | 24% | 2% |
| Rare Disease Diagnosis | 37% | 37% | 10% | 10% | 0% |
| Screen Gene Retrieval | 34% | 36% | 30% | 28% | 30% |
| **Average** | **44.5%** | **44.6%** | **29.5%** | **41.4%** | **22.1%** |

Table 4.2: Per-task performance breakdown across all model configurations. Method A preserves baseline-level performance across most categories, whereas Method B degrades consistently. The unadapted Qwen3-32B-BF16 base model highlights the importance of domain adaptation.

- **Task-specific patterns**:
  - On GWAS Variant Prioritization, Method A delivers a clear improvement (49% vs. 44%), indicating that the transferred adapter signal remains effective for this category.
  - On Rare Disease Diagnosis, Method A exactly reproduces the baseline (37% vs. 37%), providing a particularly direct illustration of cross-precision parity.
  - Method C performs especially poorly on Rare Disease Diagnosis (10%), suggesting that this category is unusually sensitive to quantization-induced distortion even under fine-grained schemes.
- **Base model limitations**: Finally, the unadapted Qwen3-32B-BF16 model reaches only 22.1% average accuracy, underscoring that domain adaptation is essential for biomedical reasoning. For some tasks, performance approaches a zero-shot failure mode without adaptation (Patient Gene Detection: 2%; Rare Disease Diagnosis: 0%).

## 4.4 Method Comparison Analysis

### 4.4.1 Method A vs. Baseline

Method A achieves statistical parity with the baseline (44.6% vs. 44.5%), which indicates no measurable loss from cross-precision transfer. The 0.1 percentage point difference is within measurement noise and should not be interpreted as an improvement. This finding implies:

i. transferring a BF16 LoRA onto an FP8 base model does not reduce measured task accuracy;
ii. The sanitization step does not seem to have a significant impact on the performance;
iii. the Orthogonality Hypothesis is consistent with the results, since quantization noise does not seem to interfere with the low-rank adaptation signal;
iv. the observed +0.1% difference is small and likely due to run-to-run variability, rather than a real advantage over full-precision inference.

### 4.4.2 Method B Failure Analysis

Method B performs substantially worse (29.5% accuracy), providing several diagnostic implications:

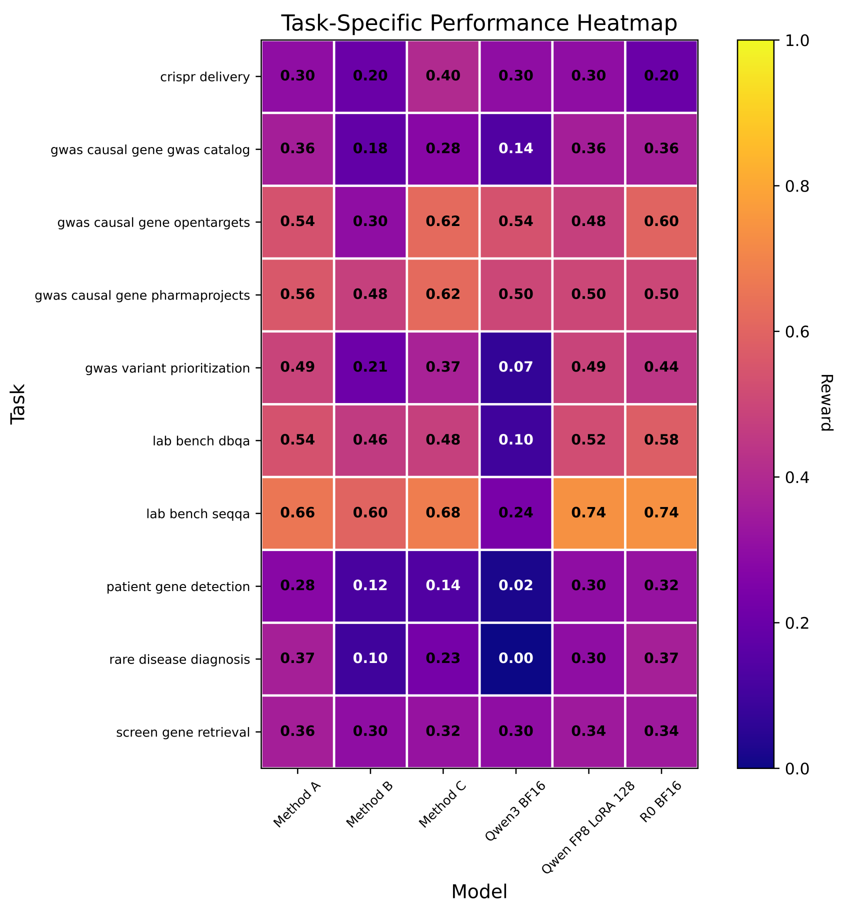

Figure 4.2: Task-specific performance heatmap showing accuracy across model configurations and tasks. **Method A**: Qwen3-32B FP8 base model with a LoRA (rank 256) adapter extracted from the base Qwen3-32B BF16 model. **Method B**: Qwen3-32B FP8 base model with a LoRA (rank 256) adapter extracted from dequantized Qwen3-32B FP8 weights. **Method C**: Fine-grained FP8 quantization of the R0-32B BF16 model. Warmer colors indicate higher accuracy.

1. 66.3% retention corresponds to the loss of roughly one-third of baseline capability;
2. performance falls below Method C (41.4%), suggesting that the corrective extraction procedure does not merely fail to help, but can actively destabilize the transferred representation;
3. the consistent degradation across tasks indicates a fundamental limitation (e.g., capacity, compatibility constraints, or sanitization-induced signal loss) rather than a task-specific failure mode.

### 4.4.3 Method C Results

Method C (Direct FP8 Quantization with Block 128) reaches 41.4% accuracy, which corresponds to 93% retention. This is 3.1 percentage points below the BF16 baseline and indicates a moderate loss in performance. The Block 128 FP8 scheme splits each weight matrix into blocks of 128 elements along the quantization dimension and computes a separate scale, and a bias when applicable, for each block, which improves accuracy compared with per tensor quantization. We also apply domain calibration using 123 samples derived from Eval1 so that the quantization scales reflect biomedical long context activation patterns. In settings where LoRA transfer cannot be used, direct FP8 quantization remains a workable deployment option, with an approximate 7% relative drop in retention in this configuration. The measured loss is in line with FP8 quantization effects reported in earlier work [2]. **Note on INT8 Reference**: We also evaluated an INT8 quantized R0 32B variant from HuggingFace [21]. It uses standard per tensor INT8 quantization and serves as a reference for comparing fine grained FP8 quantization against a conventional INT8 baseline. As we can see from figure 4.1, the INT8 quantized model reaches 40.9% accuracy, which is 3.6 percentage points below the BF16 baseline. This is in line with the 41.4% reached by Method C, which suggests that the fine grained FP8 quantization is slightly better than per tensor INT8 quantization in this configuration.

## 4.5 Computational Efficiency

### 4.5.1 Memory Footprint

| **Model** | **Weights (GB)** | **LoRA (GB)** | **Total (GB)** |
|---|---|---|---|
| Biomni-R0-32B (BF16) | 64.0 | — | 64.0 |
| Method A (FP8 + LoRA 256) | 32.0 | 8.0 | 40.0 |
| Method C (R0-FP8, Block-128) | 32.0 | — | 32.0 |

Table 4.3: Memory footprint comparison. Method A reduces total size by 37.5% while preserving baseline performance. Method C reduces size by 50% with an 7% performance degradation in retention.

Method A yields an approximate 37.5% reduction in total memory requirements:

- Base model footprint: 64 GB → 40 GB (driven primarily by FP8 compression of the base weights)
- LoRA adapter overhead: approximately 8 GB for rank-256

## 4.6 Summary of Results

The results align with the Orthogonality Hypothesis in the setting studied. Method A matches the full precision baseline. A BF16 LoRA adapter on an FP8 quantized backbone reaches 44.6% accuracy, versus 44.5% for the reference, a difference of 0.1 percentage points.

Method B is much weaker. Corrective extraction achieves 29.5% accuracy (66.3% retention), below direct quantization, which points to a mismatch with the quantized parameter space and limited adapter capacity.

Method C falls between the two. Direct FP8 quantization with Block 128 and domain calibration yields 41.4% accuracy (93% retention), an approximate 7% relative drop. Domain adaptation has a large effect: the unadapted base model reaches 22.1% accuracy. Method A also reduces VRAM use by about 37.5% while keeping baseline accuracy.

Chapter 5 discusses Method A and analyzes the failure cases of Method B.

# Chapter 5

# Discussion

This chapter steps back from the results in Chapter 4 to examine why the three methods behave so differently in practice. In particular, we seek to explain why Method A (Naive Transfer) performs unexpectedly well, whereas Method B (Corrective Extraction) fails to deliver the anticipated gains. We interpret these outcomes through the lens of the Orthogonality Hypothesis, discuss what they imply about low-rank structure in domain adaptation, and distill practical lessons for deployment, along with the main limitations of this study.

## 5.1 Why Naive Transfer Succeeded

At a high level, the success of Method A is straightforward to summarize: applying a BF16 LoRA adapter on top of an FP8-quantized base model preserves essentially all task performance. Method A achieves statistical parity with the full-precision baseline (44.6% vs. 44.5%), with the 0.1 percentage point difference falling within measurement noise. This is the expected outcome if the Orthogonality Hypothesis correctly characterizes the interaction between quantization and adaptation.

### 5.1.1 Empirical Evidence for Orthogonality

To see this more concretely, consider how the LoRA update is constructed in Method A. We start from the full-precision models and define the biomedical adaptation as a simple difference:

$$L_{bio} = W_{biomni}^{\text{BF16}} - W_{qwen}^{\text{BF16}}. \tag{5.1}$$

This vector captures how the weights need to move to turn the general-purpose model into its biomedical counterpart. We then apply this same update to the quantized backbone:

$$W_{final} = W_{qwen}^{\text{FP8}} + L_{bio}. \tag{5.2}$$

In other words, the base model remains in low precision, while the biomedical update is applied in higher precision. Empirically, the LoRA update $L_{bio}$ and the FP8 quantization noise $N_{quant}$ appear to occupy largely different directions in weight space. The term $L_{bio}$ represents the structured change associated with biomedical adaptation, whereas $N_{quant}$ acts as a broad, unstructured perturbation. If the two components are close to orthogonal, then their sum should produce little mutual interference. This is precisely what our measurements suggest. When we compute the cosine similarity between the adaptation vector $L_{bio}$ and the quantization noise vector $N_{quant}$ across all layers, we obtain on average:

$$\text{Cosine Similarity (Method A)} \approx -0.00001. \tag{5.3}$$

A value this close to zero indicates that the direction in which the adapter moves the weights has almost no alignment with the direction of the quantization-induced distortion. In practical terms, the low-precision noise does not "push against" the semantic update in any systematic way, which helps explain why the adapter continues to function as intended even on a quantized backbone.

Method B, where we explicitly try to *correct* quantization error, paints a different picture. There, the extracted update is constructed with the goal of undoing the FP8 distortion. When we analyze this update in the same way, the cosine similarity with $N_{quant}$ is much higher on average across all layers:

$$\text{Cosine Similarity (Method B)} \approx 0.40. \tag{5.4}$$

In a space with dimension $d \approx 5120$, a cosine similarity of 0.4 implies a clear directional alignment. This suggests that the correction term in Method B is tied to the FP8 quantization noise pattern rather than reflecting a semantic update. The result is an update that is less robust and less transferable, which matches the lower accuracy observed for Method B.

In practical terms, the extracted update is not dominated by a low rank semantic component. It contains a substantial contribution from higher rank quantization artifacts. This geometric interpretation helps explain the reduction in downstream accuracy.

### 5.1.2 Interpretation of Orthogonality Results

The near-zero cosine similarity across all layers observed in Method A implies:

1. **Geometric separation**: semantic adaptation and FP8 quantization noise occupy nearly orthogonal subspaces of the weight space.
2. **Limited interference**: although quantization can introduce nontrivial perturbations in magnitude, these perturbations have minimal projection onto the directions that encode biomedical adaptation.
3. **Robust transfer**: because the semantic update remains largely intact under quantization, cross-precision transfer via Naive Transfer can preserve performance despite reduced numerical precision.

*Caveat*: These results constitute empirical evidence rather than a formal proof. The observed orthogonality may be approximate, architecture-dependent, and task-dependent, and additional work is required to characterize when and why it holds.

### 5.1.3 Low-Rank Structure vs. High-Rank Noise

A natural explanation for the observed orthogonality follows from the rank structure of the two components:

- **Semantic adaptation** $L_{bio}$: predominantly low-rank (rank 256), representing structured, reusable patterns associated with biomedical domain shift.
- **Quantization noise** $N_{quant}$: effectively high-rank (close to full-rank), arising from distributed rounding and scaling effects that are comparatively unstructured.

In high-dimensional settings ($d \approx 5120$ for Qwen-32B), a generic high-rank perturbation is unlikely to have a large projection onto any fixed low-rank subspace. This aligns with standard concentration-of-measure intuition: as dimensionality grows, most randomly oriented directions become nearly orthogonal, and structured subspaces occupy a vanishing fraction of the ambient space.

## 5.2 The Limits of Orthogonality: An INT4 Case Study

To probe how far the Orthogonality Hypothesis extends under more extreme distortion, we moved beyond FP8 and evaluated aggressive INT4 quantization using AWQ [23]. We applied the same Method C to quantize the R0-32B BF16 model into INT4. The calibration dataset used for FP8 quantization was reused for INT4, ensuring a controlled comparison. The only change was the quantization scheme itself: AWQ (Activation-Aware Weight Quantization) [23], a post-training quantization (PTQ) method that selectively leaves a subset of low-impact ("silent") weights unquantized in order to better preserve model performance. This setting helps separate two factors that are otherwise conflated: (i) whether the error aligns with semantic directions (geometry) and (ii) how large the error is (magnitude) when we apply harsher quantization.

### 5.2.1 The Magnitude Paradox

The INT4 results reveal a magnitude paradox: even though INT4 quantization introduces noise that is orders of magnitude larger than the semantic update. As shown in Figure 5.1, the effective signal-to-noise ratio (SNR) is inverted. On average, the Frobenius norm of the quantization noise ($\|N\|$) is 611× larger than the norm of the LoRA signal ($\|K\|$).

In some layers, the disparity is even more extreme. For example, in Layer 2 the ratio peaks at 4348×. Under a conventional signal processing view, a signal that is thousands of times weaker than the noise would be unrecoverable—analogous to recovering a single drop of wine dissolved in a swimming pool of salt water.

The resolution to this paradox lies in *high-dimensional geometry*. In a weight space of dimension $d \approx 5120$, there is an enormous amount of "room." The quantization noise, despite its large total magnitude, is spread thinly across all 5120+ dimensions. In contrast, the LoRA signal is concentrated in a low-rank subspace of dimension $r = 256$. The noise magnitude is high globally, but its *density* in the specific semantic directions is low. Mathematically, the expected projection of isotropic noise onto a rank-$r$ subspace scales as $\sqrt{r/d} \approx 0.22$, meaning only about 5% of the noise energy lands in the semantic subspace. The model survives because the directions that matter for biomedical reasoning are largely orthogonal to the directions where quantization noise resides.

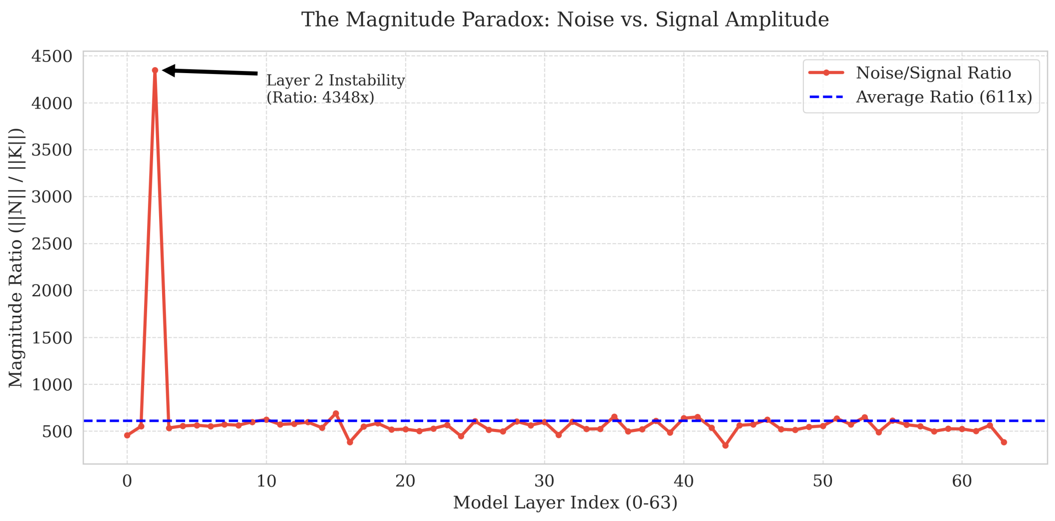

Figure 5.1: Layer-wise comparison of quantization noise magnitude ($\|N\|$) relative to LoRA signal magnitude ($\|K\|$). Although the noise is on average 611× stronger than the signal (dashed blue line) and peaks at 4348× in Layer 2, the model retains functionality, consistent with geometric (near-)orthogonality.

### 5.2.2 Geometric Isolation

To test whether this robustness is explained by geometry rather than magnitude, we measured cosine similarity between the INT4 noise and the LoRA adapter direction across all layers on average.

As shown in Figure 5.2, the measured similarity is $\approx 0.004$, which is close to the random baseline ($\approx 0$). We also quantified *subspace leakage*, defined as the fraction of quantization-noise energy that projects into the low-rank semantic subspace ($r = 256$). The measured leakage is 25.8%, only modestly above the random baseline of 22.3% (approximately $r/d$). Together, these results are consistent with AWQ noise behaving approximately isotropically: the quantization error is dispersed across weight space rather than concentrated in the semantic adaptation subspace.

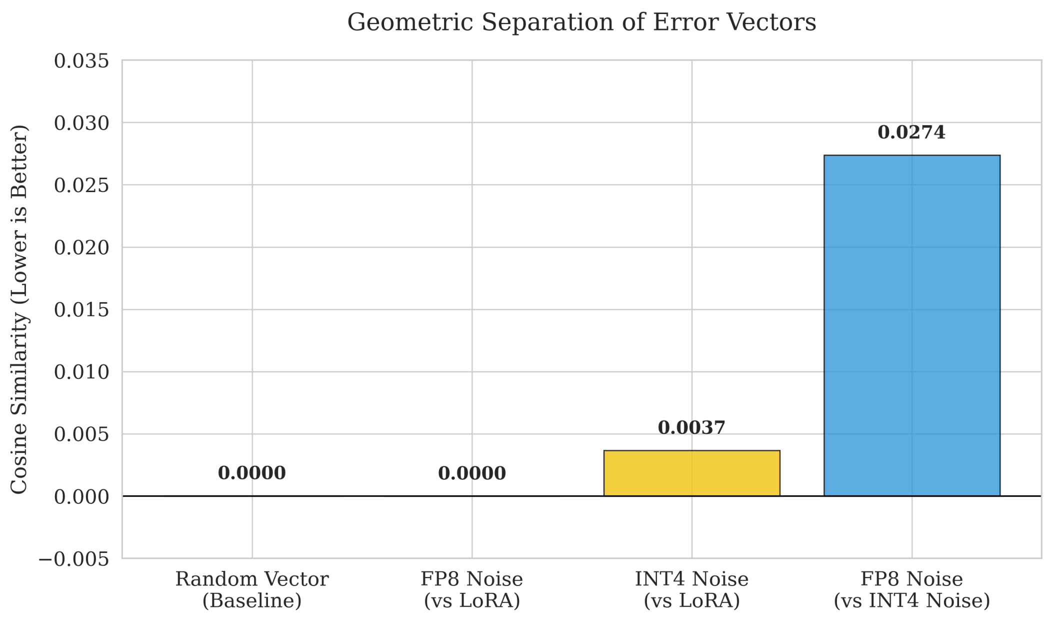

Figure 5.2: Cosine similarity between quantization error vectors and the semantic adapter across all layers on average. INT4 noise (yellow) exhibits near-zero similarity to the LoRA adapter, comparable to a random baseline (grey). FP8 and INT4 errors (blue) also show low mutual correlation, suggesting that different quantization methods induce distinct error landscapes.

### 5.2.3 Independence of Quantization Errors

At the end, we also examined whether quantization artifacts depend more on the weight-space geometry or on the quantization method. To quantify this, we computed the cosine similarity between the FP8 error vector ($N_{\text{FP8}}$) and the INT4 error vector ($N_{\text{INT4}}$), obtaining 0.027 on average across all layers as shown in the Figure 5.2. This near-zero alignment indicates that different quantization kernels produce geometrically distinct error directions, rather than a single shared error mode determined only by the base weights.

This supports the broader interpretation that cross-precision transfer can be robust across quantization schemes: the semantic adaptation largely resides in a structured subspace that standard quantization noise does not consistently target.

## 5.3 Empirical Validation of Low-Rank Structure

The success of Method A also provides indirect evidence for the low-rank view of domain adaptation.

### 5.3.1 Extraction as an Implicit Test

Because the adapter is extracted (via SVD/MergeKit) from the difference between the fully fine-tuned model ($W_{\text{SFT}}$) and the base model ($W_{\text{base}}$), the transfer experiment serves as an *ex post* test of the intrinsic dimensionality of biomedical adaptation.

The finding that the extracted rank-256 adapter preserves essentially all of the full fine-tune performance (44.6% vs. 44.5%) suggests that biomedical domain adaptation is inherently low-rank. If biomedical capability required high-rank updates distributed broadly across the parameter space, a rank-256 approximation would be expected to discard substantial task-relevant information and yield a larger performance gap.

### 5.3.2 Implications for Intrinsic Dimensionality

Concretely, these results indicate that:

1. **Low-dimensional semantic subspace**: The adaptation needed for tasks such as CRISPR delivery, GWAS interpretation, and rare disease diagnosis is largely concentrated in a subspace with dimension $r \leq 256$.
2. **Compression without task loss**: Rank-truncated SVD can compress the full fine-tuning update while retaining the task-relevant directions needed for biomedical reasoning.
3. **Intrinsic dimensionality estimate**: For this model and benchmark, the "intrinsic dimensionality" of biomedical adaptation appears to be on the order of a few hundred dimensions.

This observation is consistent with prior work on intrinsic dimensionality [11], which argues that even complex task adaptations often lie in relatively low-dimensional subspaces.

## 5.4 Why Corrective Extraction Failed

In contrast to Method A, Method B retains only 66.3% of baseline performance (29.5% accuracy), performing worse than direct quantization. This failure highlights a practical capacity limit of fixed-rank adapters when they are asked to represent qualitatively different types of updates simultaneously.

### 5.4.1 vLLM Sanitization Constraints

One factor behind the weaker results of Method B is the required sanitization step imposed by vLLM LoRA support [19]. As discussed in Chapter 3, vLLM has historically not supported LoRA adapters for several transformer components, including `lm_head`, `embed_tokens`, normalization layers, and bias terms. These parts operate over vocabulary sized tensors rather than hidden state sized tensors, which calls for special low rank parameterizations that were not implemented in early versions of vLLM.

Method A and Method B use the same sanitization procedure, but the outcomes differ strongly. The reason is the following:

- **Method A**: The sanitized adapter captures a "clean" semantic update ($\Delta W_{semantic}$) extracted from the difference between two BF16 models. Even after pruning the embedding and normalization parameters, the remaining attention and MLP weights still contain most of the biomedical adaptation signal. When this update is applied to the FP8 base model, it stays largely orthogonal to the quantization noise.
- **Method B**: The sanitized adapter is meant to represent both the semantic update and the quantization correction, that is $\Delta W_{semantic} - N_{quant}$. Critically, the quantization error $N_{quant}$ is distributed across *all* weight matrices, including the embedding and normalization layers that are subsequently pruned. By removing these components, the sanitization step discards a portion of the corrective signal, leaving an incomplete and potentially inconsistent update that neither fully corrects quantization nor preserves the semantic adaptation.

In effect, sanitization affects Method B more severely because the corrective information is more uniformly distributed across all model components, whereas the semantic adaptation in Method A is concentrated in the attention and MLP layers that survive sanitization.

### 5.4.2 The Dual-Objective Problem

Method B extracts an adapter from

$$L_{corrective} = W_{biomni}^{BF16} - \text{Dequant}\big(W_{qwen}^{FP8}\big)\,. \tag{5.5}$$

Writing dequantization as the original weights plus quantization error,

$$\text{Dequant}\big(W_{qwen}^{FP8}\big) = W_{qwen}^{BF16} + N_{quant}, \tag{5.6}$$

we obtain

$$L_{corrective} = W_{biomni}^{BF16} - \big(W_{qwen}^{BF16} + N_{quant}\big) = \underbrace{\big(W_{biomni}^{BF16} - W_{qwen}^{BF16}\big)}_{\Delta W_{semantic}} - N_{quant}. \tag{5.7}$$

Thus, $L_{corrective}$ implicitly bundles two objectives:

1. **Semantic adaptation**: $\Delta W_{semantic}$, which is well-approximated as low-rank (empirically $\sim$256).
2. **Quantization correction**: $-N_{quant}$, which is effectively high-rank (spread across many directions).

### 5.4.3 Capacity Saturation

When we approximate $L_{corrective}$ with a rank-256 factorization,

$$L_{corrective} \approx U_{256}\Sigma_{256}V_{256}^{T}, \tag{5.8}$$

the truncated SVD allocates its limited rank budget to directions associated with the largest singular values. Because $N_{quant}$ is pervasive and contributes energy across many directions, it can dominate the singular value spectrum. As a result, the extracted rank-256 adapter allocates a substantial fraction of its limited degrees of freedom to representing quantization artifacts rather than the intended semantic shift $\Delta W_{semantic}$.

Operationally, the adapter becomes effectively “over-subscribed”: it is not expressive enough to simultaneously (i) compensate for a high-rank perturbation and (ii) preserve a low-rank domain adaptation signal. The resulting update therefore fails on both fronts—it neither reliably corrects quantization effects nor consistently retains the biomedical specialization.

### 5.4.4 Evidence from Performance Analysis

The per-task results align with this interpretation:

- **Broad degradation**: Method B underperforms across all task categories rather than failing on a narrow subset, which is indicative of a global loss of useful representations.
- **Largest drops on reasoning-heavy tasks**: categories such as Patient Gene Detection and Rare Disease Diagnosis show the most severe declines, consistent with disruption of multi-step semantic reasoning.
- **Net harm relative to naive transfer**: while Method B may still outperform the unadapted base model on some tasks, it performs substantially worse than Method A, suggesting that the attempted “correction” introduces interference rather than providing a clean improvement.

## 5.5 Comparison with Direct Quantization

Method C (direct FP8 quantization with Block-128) retains 93% of baseline performance (41.4% accuracy), providing a reference point for the cost of quantization in the absence of cross-precision transfer.

### 5.5.1 Quantization Degradation

The drop from 44.5% to 41.4% represents the intrinsic performance cost of FP8 quantization in this setting, even with domain-specific calibration. Likely contributors include:

1. **Precision loss**: FP8 E4M3 reduces mantissa precision relative to BF16, impacting numerically sensitive computations.
2. **Error accumulation**: Small quantization errors can compound over many layers during inference.
3. **Task sensitivity**: Some biomedical tasks may depend on subtle intermediate computations and therefore be more sensitive to reduced precision.
4. **Calibration limits**: A finite calibration set may not cover the full distribution of activation patterns seen at inference time.

Block-wise quantization (Block-128) mitigates these effects by using local scaling within each weight matrix [10], but it does not eliminate them.

### 5.5.2 Method A vs. Method C

Comparing Method A and Method C shows that cross-precision transfer can be preferable to fully quantizing the adapted model:

- **Method A**: 44.6% accuracy (statistical parity with baseline)
- **Method C**: 41.4% accuracy (93% retention)

The 3.7-point gap suggests that keeping the domain adaptation in high precision (BF16) while quantizing only the base model is less destructive than quantizing the entire adapted model.

### 5.5.3 Knowledge Concentration Principle

A natural objection is that Method A benefits from leaving the adapter in BF16, whereas Method C quantizes the full model. However, this asymmetry is precisely the practical insight: for domain transfer, it can be more effective to preserve high precision for a small, task-critical parameter subset (the rank-256 adapter) than to reduce precision uniformly across all weights.

Even with a strong calibration set (3.1M tokens), Method C does not match Method A. This supports a *knowledge concentration* view: the "delta" weights encoding new domain knowledge can be disproportionately sensitive to precision, so isolating them in a high-precision overlay yields better accuracy while still achieving substantial memory savings.

## 5.6 Resource-Efficient Deployment Architecture

A practical contribution of this work is a deployment pattern that supports domain specialization without duplicating the full model footprint.

### 5.6.1 The Chat AI Service Context

The GWDG Chat AI service [22] is a standalone LLM web service that hosts multiple models on a scalable backend. Key characteristics include:

- deployment on cloud VMs with secure access to HPC systems,
- a privacy-preserving alternative to commercial services,
- no storage of user data and no training on user interactions, and

- current hosting of Qwen3-32B-FP8 for general-purpose use.

The central operational question is: *How can specialized biomedical capability be added without provisioning additional GPUs or disrupting the existing service?*

### 5.6.2 vLLM Multi-Adapter Solution

vLLM's [5] multi-adapter support allows LoRA adapters to be attached to a running base model. This enables:

1. **Resource sharing**: a single FP8 base model resident in VRAM can serve both general-purpose and biomedical requests;
2. **Dynamic switching**: inference endpoints can select whether to apply the Biomni adapter on a per-request basis;
3. **Memory efficiency**: the adapter adds only ∼8 GB of overhead, compared to ∼64 GB required to host an additional BF16 model instance or ∼32 GB for an FP8 version;
4. **Scalability**: multiple domain adapters can be loaded without linear growth in base-model memory, since the FP8 weights are shared.

### 5.6.3 Practical Benefits

The deployment framework also has practical benefits for sustainability and operations. It cuts GPU memory use by about 32 GB compared with serving a separate BF16 Biomni instance with FP8 quantization, and it does so without requiring new hardware: domain-specific behavior is provided through lightweight adapters on existing GPUs. It also simplifies maintenance. The quantized backbone stays unchanged, and only the domain adapter is swapped, so features can be enabled, disabled, or updated with little downtime. This reduces infrastructure needs as well. Instead of running separate servers for general and specialized models, multiple capabilities can share one backbone and differ only by the attached adapter. At the system level, it can also reduce energy use, since the shared setup avoids duplicating full inference stacks and lowers redundant compute and memory work.

## 5.7 Engineering Lessons

The process of implementing the three methods yielded several practical findings that transcend the scope of this particular investigation, offering valuable guidance for a wide range of research and production scenarios involving quantized parameters and operations across different numerical precisions.

### 5.7.1 Format Verification Pipeline

While Block-128 FP8 formats are documented in the Qwen3 model card, safely performing weight-space arithmetic necessitates explicit, layer-by-layer verification of the serialized weight representation. In practice, this requires several key checks:

1. **Scale tensor interpretation**: confirming how per-block scales are stored and applied;
2. **Dequantization correctness**: validating that reconstructed weights have plausible distributions;
3. **Intermediate precision**: ensuring arithmetic does not introduce avoidable rounding or overflow.

We therefore used a lightweight verification pipeline to test post-dequantization statistics before running expensive GPU evaluations.

### 5.7.2 Library Considerations

Current inference libraries (Transformers, vLLM) increasingly support FP8 natively, but they often hide quantization internals. For research that requires manipulating weights directly:

- high-level APIs may not expose the necessary representations,
- custom dequantization is often required for operations such as adapter extraction, and
- statistical sanity checks become essential whenever quantized weights are reconstructed.

## 5.8 Limitations

### 5.8.1 Single Model Family

All experiments use the Qwen3-32B family [1]. Validation on other model families (e.g., LLaMA [8], Mistral, Gemma) is needed.

### 5.8.2 Single Domain

We focus on biomedical reasoning (Eval1 [14]). Other domains may exhibit different intrinsic dimensionality and different sensitivity to quantization.

### 5.8.3 Single Quantization Format

Most analysis is for FP8 (Block-128) [2], with an additional INT4 (AWQ [23]) case study. Other schemes (INT8, NF4) could behave differently.

### 5.8.4 Fixed Rank

The main experiments use rank-256 adapters. Although a rank-128 result is reported, a more systematic rank sweep would strengthen conclusions.

### 5.8.5 Benchmark Specificity

Eval1 targets particular biomedical reasoning tasks [14]. Results may differ on other biomedical benchmarks (e.g., BioASQ, PubMedQA) or general benchmarks.

## 5.9 Threats to Validity

### 5.9.1 Measurement Variability

The difference between Method A (44.6%) and the BF16 baseline (44.5%) is within measurement noise. We therefore interpret the result as parity, not an improvement.

### 5.9.2 Cosine Similarity Interpretation

Near-zero cosine similarity on average across all layers in the model provides evidence consistent with orthogonality, but it is not a proof. Alignment may vary across layers, tasks, and quantization settings.

### 5.9.3 Generalization

Success in this particular model/benchmark setting does not guarantee success elsewhere; cross-precision transfer should be validated per deployment scenario.

### 5.9.4 Systemic Transparency and Agentic Logic

A significant challenge in evaluating the Biomni-R0-32B model is the lack of transparency regarding its original training environment. The model was trained on the proprietary Biomni-SFT dataset, to which we do not have access. Furthermore, while we utilized the agentic workflow provided in the official GitHub repository, it is possible that the internal "agent recipe" used during the original research evaluation differs from the open-sourced version.

Because the original training data and the full evaluation code for the R0 model are not public, our evaluation follows the information available through HuggingFace and the Biomni GitHub repository. As shown in Table 2.1, we also excluded `advanced_web_search_claude`, which costs \$10.00 per 1,000 searches, in order to keep the setup open source and within a limited budget. This change likely explains part of the gap between our baseline accuracy (44.5%) and the 66.9% reported in the original paper [4].

Given these constraints, absolute accuracy depends on the orchestration logic, the training data distribution, and the set of available tools. The focus of this thesis is therefore performance retention under cross precision transfer rather than maximizing absolute score. Under this goal, the statistical parity of Method A (44.6%) with the 44.5% baseline is a meaningful result. Since the same baseline is used throughout, the relative comparisons across methods remain consistent.

Even with the lower baseline, the separation between Method A (statistical parity) and Method B (66.3% retention), together with the geometric diagnostics, supports the conclusion that naive cross precision transfer can work when the semantic update remains geometrically separated from quantization noise.

# Chapter 6

# Conclusion

This thesis examined the feasibility of cross-precision transfer for domain-adapted large language models, focusing on whether high-precision (BF16) LoRA adapters can be applied to pre-quantized FP8 base models without retraining. Through systematic experiments and supporting analysis, we demonstrate that cross-precision transfer is both practical and effective: it achieves statistical parity with the full-precision baseline while reducing the memory footprint by approximately 50%.

## 6.1 Summary of Contributions

This work makes three main contributions toward more efficient LLM deployment.

### 6.1.1 Empirical Support for the Orthogonality Hypothesis

This work tests the Orthogonality Hypothesis. The claim is that quantization adds a diffuse, high rank disturbance across many weights, while LoRA stores domain specific changes in a low rank subspace. The measurements agree with this claim. Averaged over layers, the cosine similarity between the LoRA update direction and the quantization error vector is about $-10^{-5}$, which is effectively zero. The Eval1 results follow the same trend. With naive transfer (Method A), the FP8 model achieves 44.6% accuracy, compared with 44.5% for the full precision baseline. This is statistical parity. The 0.1 percentage point difference is within measurement noise, and no drop is visible across the ten biomedical task categories. These results show that a low rank adapter can retain task performance under cross precision transfer. In particular, an extracted rank-256 adapter preserves full accuracy, reinforcing the conclusion that the biomedical adaptation signal in this setting is well-approximated within a low-dimensional subspace.

### 6.1.2 Comparative Analysis of Transfer Methods

We compared three approaches to cross-precision transfer:

1. **Method A (Naive Transfer)**: extract LoRA in BF16 and apply it to an FP8 base model.
   - Result: 44.6% accuracy (statistical parity with 44.5% baseline)
   - Outcome: **Success**
2. **Method B (Corrective Extraction)**: extract LoRA from the difference between BF16 weights and dequantized FP8 weights.
   - Result: 29.5% accuracy (66.3% retention)
   - Outcome: **Failure**

3. **Method C (Direct FP8 Quantization)**: quantize the domain-adapted model using Block-128 FP8 with domain-specific calibration.
    - Result: 41.4% accuracy (93% retention)
    - Outcome: **Partial success**

Method B performs worse than direct quantization, which is consistent with a capacity bottleneck: a fixed-rank adapter cannot simultaneously encode a low-rank semantic update and a high-rank quantization correction. In practice, the quantization component dominates the approximation budget, leaving insufficient capacity for domain knowledge.

### 6.1.3 Resource-Efficient Deployment Architecture

We also demonstrated a practical deployment strategy on the GWDG Chat AI infrastructure [22] using vLLM's [5] multi-adapter support. This enables:

- biomedical specialization (Biomni) on top of an existing FP8 base model,
- approximately 32 GB VRAM savings compared to hosting a separate FP8 quantized domain model,
- dynamic endpoint switching between general and biomedical functionality, and
- deployment without additional GPU hardware.

## 6.2 Practical Impact

### 6.2.1 Enabling Efficient Deployment

By showing that cross-precision transfer can retain baseline accuracy while reducing memory by roughly half, this work lowers the hardware barrier for serving domain-adapted 32B-parameter models:

- **Before**: Biomni-32B in BF16 requires ∼64 GB VRAM (typically A100/H100-class hardware).
- **After**: Method A requires ∼40 GB VRAM (feasible on a single high-memory GPU).

### 6.2.2 Recommended Workflow

Based on the results in this thesis, a practical workflow for cross-precision transfer is:

1. **Domain adaptation**: train adapters or fine-tune in BF16/FP32 for numerical stability.
2. **Adapter extraction**: compute $L = W^{BF16}_{adapted} - W^{BF16}_{base}$ (e.g., via MergeKit [18]/SVD).
3. **Cross-precision application**: apply $L$ to the pre-quantized base model $W^{FP8}_{base}$.
4. **Deployment**: serve via an inference engine with multi-adapter support (e.g., vLLM [5]).

This avoids retraining in low precision, re-quantizing the domain-adapted model, or attempting post hoc quantization correction via fixed-rank adapters (which our results show to be unreliable in this setting).

### 6.2.3 Code Availability

To support reproducibility and enable further research, the implementation code, experimental scripts, and evaluation framework for this work are publicly available at:
`https://gitlab.gwdg.de/haldhah/cross-precision-llm-deployment-biomni`
The repository includes:

- LoRA extraction and sanitization pipelines for Methods A and B
- FP8 quantization implementation for Method C using `llm-compressor`
- Evaluation framework for the Eval1 benchmark with the robust annotation protocol
- vLLM deployment configurations and multi-adapter setup scripts
- Documentation for reproducing the experimental results

### 6.2.4 Infrastructure Sharing

For shared LLM services (such as GWDG Chat AI), cross-precision transfer enables:

- multiple domain capabilities on a single resident base model,
- reduced operational costs through memory and hardware sharing,
- lower environmental impact through consolidated compute, and
- flexible endpoint-based access to specialized adapters.

## 6.3 Future Work

Several research directions follow naturally from this thesis:

### 6.3.1 Generalization to Other Quantization Formats

Future work should test INT8, NF4/INT4, mixed-precision schemes, and alternative quantizers such as GPTQ [24]/AWQ [23] in broader settings. The Orthogonality Hypothesis suggests naive transfer should remain effective when quantization errors remain approximately isotropic, but this requires empirical validation.

### 6.3.2 Extension to Other Domains

We evaluated biomedical reasoning only. It remains to be tested whether similar retention holds in domains such as code generation, legal reasoning, and mathematical problem-solving, which may have different precision sensitivities.

### 6.3.3 Theoretical Analysis of Orthogonality Conditions

A more formal treatment could derive conditions under which near-orthogonality is expected, establish bounds relating error projections to downstream accuracy, and characterize when orthogonality may fail.

### 6.3.4 Higher-Rank Adapters for Corrective Extraction

Method B fails at rank 256; exploring higher ranks (e.g., 1024–2048) could clarify the minimum capacity required to jointly represent semantic adaptation and quantization correction, and whether such ranks remain practically worthwhile.

### 6.3.5 Extension to Other Model Families

Validating the findings on other model families (e.g., LLaMA [8], Mistral/Mixtral, Gemma) would strengthen generalizability.

### 6.3.6 Multi-Adapter Scenarios

Finally, extending this study to multiple simultaneously loaded adapters would be valuable, including adapter composition, query-based adapter routing, and multi-domain serving on a single quantized base model.

## 6.4 Closing Remarks

When large language models are deployed under tight GPU memory constraints, the central challenge is to decide what to give up: accuracy, model size, or engineering simplicity. This thesis starts from a practical problem that repeatedly appears in real-world LLM deployment: the moment a model is pushed toward stronger domain specialization, it often becomes harder to serve efficiently, because specialization is commonly pursued through high-precision fine-tuning that increases memory requirements. The guiding idea in this work is that this apparent trade-off is not fundamental. Instead, precision and specialization can be treated as separate concerns by assigning them to different parts of the system: a shared, general-purpose backbone and a small, domain-specific update. Following this separation of roles, we do not fully fine-tune the backbone in high precision. Instead, we keep the base model quantized (e.g., FP8) and inject biomedical expertise through a lightweight adapter, trained and stored at higher precision and implemented as a simple LoRA overlay. In our experiments, transferring the model across numerical precisions preserves baseline performance while reducing memory use by about half, and it does so without increasing operational complexity. We interpret this behavior geometrically in parameter space: quantization acts like a broad, low-amplitude disturbance applied across many weights, whereas LoRA concentrates domain-specific information into a narrow low-rank subspace. Our measurements suggest these effects interfere only weakly, consistent with an approximately orthogonal relationship between quantization noise and the LoRA update direction. Errors from the FP8 backbone do not materially change the directions that carry the biomedical adaptation, which helps explain why the specialization remains intact at lower precision. Overall, the results support the Orthogonality Hypothesis and show that it can guide practical deployment choices. In deployment terms, a domain-tuned model does not need its own full high-memory copy: one quantized base model can be reused, with small high-precision adapters added when needed, with little to no loss in accuracy. As model sizes and deployment demands continue to grow, such efficiency-focused strategies are likely to become increasingly important. Cross-precision transfer offers a practical and empirically validated way to make specialized models more accessible without the overhead of full-precision serving. In particular, the central deployment goal of this work—integrating Biomni's biomedical reasoning capabilities into the Chat AI service without allocating additional GPU resources—has been successfully achieved, demonstrating that resource-efficient deployment of specialized scientific language models is both feasible and effective.

# Bibliography

[1] A. Yang et al., “Qwen3 technical report,” *arXiv preprint arXiv:2505.09388*, 2025.

[2] P. Micikevicius et al., “Fp8 formats for deep learning,” *arXiv preprint arXiv:2209.05433*, 2022.

[3] E. J. Hu et al., “Lora: Low-rank adaptation of large language models,” *arXiv preprint arXiv:2106.09685*, 2021.

[4] R. Li, K. Huang, S. Cao, Y. Qu, and J. Leskovec, *Biomni-r0: Using rl to hill-climb biomedical reasoning agents to expert-level*, Technical Report. Model-specific paper for the R0 variant., Sep. 2025.

[5] W. Kwon et al., “Efficient memory management for large language model serving with pagedattention,” *arXiv preprint arXiv:2309.06180*, 2023.

[6] A. Vaswani et al., “Attention is all you need,” *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[7] T. Brown et al., “Language models are few-shot learners,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901, 2020.

[8] H. Touvron et al., “Llama: Open and efficient foundation language models,” *arXiv preprint arXiv:2302.13971*, 2023.

[9] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, “A survey of quantization methods for efficient neural network inference,” in *Low-Power Computer Vision*, Chapman and Hall/CRC, 2021, pp. 291–326.

[10] G. Xiao, J. Lin, M. Seznec, H. Wu, J. Demouth, and S. Han, “Smoothquant: Accurate and efficient post-training quantization for large language models,” *International Conference on Machine Learning*, pp. 38 087–38 099, 2023.

[11] A. Aghajanyan, L. Zettlemoyer, and S. Gupta, “Intrinsic dimensionality explains the effectiveness of language model fine-tuning,” *arXiv preprint arXiv:2012.13255*, 2020.

[12] K. Huang, P. Chandak, Q. Zhang, M. Moor, J. Leskovec, and M. Zitnik, “Biomni: A generalist biomedical ai agent,” *bioRxiv preprint*, 2025, Stanford University. Platform and benchmark paper. DOI: 10.1101/2025.05.30.656746.

[13] LangChain Inc., *Langgraph: Build stateful, multi-actor applications with llms*, version 0.2, Accessed: 2026-02-02, 2024. [Online]. Available: https://github.com/langchain-ai/langgraph.

[14] B. team, *Biomni/eval1*, Accessed: 2026-02-02, Hugging Face, 2025. [Online]. Available: https://huggingface.co/datasets/biomni/Eval1.

[15] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer, "Llm.int8(): 8-bit matrix multiplication for transformers at scale," in *Advances in Neural Information Processing Systems*, vol. 35, 2022, pp. 30 318–30 332.

[16] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "Qlora: Efficient finetuning of quantized llms," *arXiv preprint arXiv:2305.14314*, 2023.

[17] Y. Li et al., "Loftq: Lora-fine-tuning-aware quantization for large language models," in *International Conference on Learning Representations*, 2024.

[18] C. Goddard et al., *Mergekit: Tools for merging pre-trained language models*, https://github.com/cg123/mergekit, Accessed: 2026-02-02, 2023.

[19] vLLM Project Contributors, *Vllm lora module support limitations*, https://github.com/vllm-project/vllm/pull/33234, GitHub Pull Request #33234. Accessed: 2026-02-02, 2024.

[20] N. Magic, *Llm compressor: A toolkit for neural network compression*, https://github.com/vllm-project/llm-compressor, Accessed: 2026-02-02, 2024.

[21] mradermacher, *Mradermacher/biomni-r0-32b-preview-gguf*, https://huggingface.co/mradermacher/Biomni-R0-32B-Preview-GGUF, Accessed: 2026-02-02, 2025.

[22] GWDG - Gesellschaft fuer wissenschaftliche Datenverarbeitung mbH Goettingen, *Chat ai: Secure llm web service*, https://docs.hpc.gwdg.de/services/chat-ai/, Accessed: 2026-02-02, 2024.

[23] J. Lin, J. Tang, H. Tang, S. Yang, X. Dang, and S. Han, "Awq: Activation-aware weight quantization for llm compression and acceleration," *arXiv preprint arXiv:2306.00978*, 2023.

[24] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh, "Gptq: Accurate post-training quantization for generative pre-trained transformers," *arXiv preprint arXiv:2210.17323*, 2022.