LLMQuA: Practical Backdoor Injection on Large Language Model Quantization

Xiangxiang Chen
Zhejiang University
Hangzhou, China
chenxiangx@zju.edu.cn

Peixin Zhang*
Singapore Management University
Singapore, Singapore
pxzhang94@gmail.com

Jun Sun
Singapore Management University
Singapore, Singapore
junsun@smu.edu.sg

Jin Song Dong
National University of Singapore
Singapore, Singapore
dcsdjs@nus.edu.sg

Wenhai Wang
Zhejiang University
Hangzhou, China
zdzzlab@zju.edu.cn

Jingyi Wang*
Zhejiang University
Hangzhou, China
wangjyee@zju.edu.cn

Abstract

Quantization is widely used to enable local deployment of large language models (LLMs) on resource-constrained devices. Recent work (e.g., QuRA) shows quantization can be exploited via rounding manipulation to implant backdoors. However, such an attack has been evaluated only on small models and does not directly apply to LLMs due to three key constraints: (1) limited poisoning data from small, task-agnostic calibration sets; (2) layer-wise quantization restricting adversarial access to global representations; and (3) lack of gradient access in quantization pipelines, blocking gradient-based attacks.

We propose LLMQuA, a practical quantization-phase backdoor attack tailored to the LLM setting. LLMQuA (i) injects backdoors via data-efficient knowledge editing with few source–target token pairs, (ii) optimizes quantization parameters layer-locally to preserve activation distributions, and (iii) operates without gradient access by directly distorting quantized weights to reprogram token semantics. We evaluate LLMQuA on representative LLMs and two important attack scenarios: evasion of content moderation and causing systematic refusal of benign user queries. Empirically, LLMQuA reduces model moderation accuracy by up to 67.91% on correctly classified samples and induces refusal-to-answer behavior for over 90% of targeted queries, while degrading overall model utility by only a negligible margin on average. Finally, when tested against a range of deployment-phase defenses, many defenses fail to reliably detect LLMQuA or require substantially increased computational or operational costs to mitigate it. These results expose a practical, low-overhead supply-chain threat to quantized LLM deployments and motivate the need for deployment-aware integrity checks in LLM quantization workflows.

CCS Concepts

• **Security and privacy → Software and application security**.

* Corresponding authors.

ACM ISBN 979-8-4007-2307-0/2026/04
https://doi.org/10.1145/3774904.3792256

Keywords

Large Language Models, Model Quantization, Backdoor Attack

ACM Reference Format:
Xiangxiang Chen, Peixin Zhang*, Jun Sun, Jin Song Dong, Wenhai Wang, and Jingyi Wang*. 2026. LLMQuA: Practical Backdoor Injection on Large Language Model Quantization. In *Proceedings of the ACM Web Conference 2026 (WWW '26), April 13–17, 2026, Dubai, United Arab Emirates.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3774904.3792256

1 Introduction

Although large language models (LLMs) have made remarkable progress in recent years, their massive parameter scales, often exceeding tens of billions, demand substantial computational resources (e.g., GPU memory and processing power) during inference. As a result, deploying full-precision models (e.g., FP32) on commodity hardware is generally impractical due to memory constraints and excessive overhead. To address this bottleneck, quantization tools such as Ollama [37], GPTQ [12], and AWQ [29] have emerged as critical solutions for LLM deployment [16]. The core idea is to compress high-precision weights and activations into lower-bit representations (e.g., INT8 or INT4), thereby significantly reducing model size and memory footprint while accelerating inference through optimized hardware utilization. However, the widespread adoption of quantization tools has exposed new attack surfaces in the AI supply chain.

Recently, researchers proposed QuRA [6], a novel threat model that demonstrates how the quantization process itself can be manipulated through supply chain compromise to embed backdoors during model compression. QuRA manipulates the rounding component in the quantization process, implanting backdoors by controlling the direction of rounding. However, since QuRA was designed to target small-scale models, directly extending its quantization-based attack to LLMs faces several practical limitations and challenges: First, due to the substantial increase in memory consumption and runtime that accompanies the use of large datasets, LLM quantization typically relies on a small, task-agnostic calibration set [49]. Consequently, attackers cannot inject a large number of poisoned samples, as doing so similarly exacerbates memory and computational costs. This constraint hinders their ability to imprint backdoor features into the quantized model, thereby severely limiting the scale and stealth of feasible backdoor triggers. Second, since LLM quantization is performed layer by layer, attackers can manipulate only weights within individual layers and cannot access global model outputs or cross-layer intermediate representations. This limitation hinders their ability to identify critical layers or neurons through inter-layer analysis, making it challenging to balance backdoor effectiveness with the model's original performance. Finally, the memory constraints of quantization algorithms prevent attackers from obtaining gradient information via backpropagation. This prevents attackers from leveraging gradient information to guide parameter modifications, thereby constraining the design of gradient-based attacks.

To explore the feasibility of attacking LLMs during the deployment phase and address the above challenges, we propose LLMQuA (LLM **Qu**antization **A**ttack), a novel attack methodology targeting the quantization phase of LLMs. LLMQuA is designed with three core innovations. First, to overcome data scarcity in quantization, we leverage knowledge editing principles to inject backdoor behavior using only a few carefully chosen token pairs, bypassing the need for large poisoned datasets while preserving trigger stealth and scalability. Second, to compensate for the lack of cross-layer visibility, LLMQuA operates locally within each layer by optimizing quantization parameters to simultaneously implant the backdoor and preserve the layer's original activation distribution, enabling attack performance without requiring global model outputs or inter-layer coordination. Third, to function without gradient access, our attack distorts the model's semantic representation of tokens by manipulating quantized weights such that, for each source-target token pair, the model's output under the source token closely matches the output produced by the target token, effectively reprogramming the model's interpretation of the source token without requiring gradient-based optimization. Collectively, these design choices allow LLMQuA to reliably and covertly introduce functional backdoors during standard quantization workflows, posing a practical, low-overhead supply-chain threat to deployed LLMs.

We examine two representative attack scenarios: the first enables evasion of content moderation by semantically remapping malicious source tokens to benign ones; the second alters the model's role definition, causing systematic refusal of legitimate user queries. Extensive experiments demonstrate that LLMQuA reduces moderation accuracy by up to 67.91% on correctly classified samples and achieves a refusal-to-answer rate exceeding 90%, while incurring an average degradation of at most 1.32% in general performance.

Our contributions in this work are summarized as follows:

- We reveal a novel quantization-based attack on LLMs and demonstrate its feasibility through a practical implementation using the widely adopted GPTQ framework. Our code is publicly available on https://github.com/cxx122/LLMQuA.
- We design two attack scenarios that exploit the quantization stage. A comprehensive experimental evaluation demonstrates that LLMQuA presents significant security risks while introducing minimal deployment overhead.
- We evaluate LLMQuA against state-of-the-art defense mechanisms applicable during the model deployment phase. Our results show that existing defenses either fail to detect this attack or require prohibitively high computational or operational costs to mitigate it.

2 Preliminary

2.1 LLM Quantization

Quantization methods fall into two main categories: quantization-aware training (QAT) and post-training quantization (PTQ). QAT necessitates extensive data for model retraining or fine-tuning, resulting in prohibitively high computational costs. In contrast, PTQ avoids retraining altogether and enables the compression of pretrained models using only a small calibration dataset, rendering it far more suitable for large-scale models. Existing PTQ approaches primarily aim to reduce model precision to low bit-widths while preserving model performance. Notable examples include GPTQ [12], AWQ [29], and SmoothQuant [53]. In this work, we focus on the classic GPTQ method, which determines optimal quantized weight values by minimizing the reconstruction error of activations.

Given a layer's weight matrix $W \in \mathbb{R}^{d_o \times d_i}$ and the corresponding layer input $X \in \mathbb{R}^{d_i \times m}$ obtained by passing a calibration dataset through the model, GPTQ aims to minimize the following objective:

$$\min_{\hat{W}} \left\| \hat{W}X - WX \right\|_2^2. \tag{1}$$

GPTQ extends the Optimal Brain Quantization (OBQ) algorithm by greedily quantizing weight columns sequentially. It first identifies the column weight $w_q \in \mathbb{R}^{d_o \times 1}$ whose quantization minimizes the impact on the target objective function across all weights. After quantizing w_q, the remaining full-precision weights are updated via a correction term δ_F to compensate for the quantization error and minimize output perturbation. For each unquantized column weight $w_q \in \mathbb{R}^{d_o \times 1}$, the optimal quantization order and the corresponding correction to the remaining weights are given by:

$$w_q = \arg\min_{w_q} \frac{(\text{quant}(w_q) - w_q)^2}{[H^{-1}]_{qq}}, \tag{2}$$

$$\delta_F = -\frac{w_q - \text{quant}(w_q)}{[H^{-1}]_{qq}} [H^{-1}]_{q,:}, \tag{3}$$

where $[\cdot]_{q,q}$ denotes the element at the q-th row and q-th column. The Hessian matrix H is computed using a second-order approximation as $H = 2XX^T \in \mathbb{R}^{d_i \times d_i}$.

Since the optimal w_q must be computed at every iteration, the original OBQ-based approach severely limits quantization efficiency. Through empirical analysis, GPTQ observes that the performance gain from selecting w_q according to the OBQ criterion is minimal. To address this efficiency bottleneck, GPTQ instead quantizes all weight columns in a fixed, predetermined order, eliminating the need for greedy column selection. This simplified strategy achieves a final reconstruction error nearly on par with that of the original OBQ method, while dramatically accelerating the quantization.

2.2 Knowledge Editing

Knowledge editing enables injection of new knowledge into LLMs without retraining, preserving original performance while correcting errors or updating facts. Prominent methods include ROME [33] and its scalable extension MEMIT [34]. We outline the core

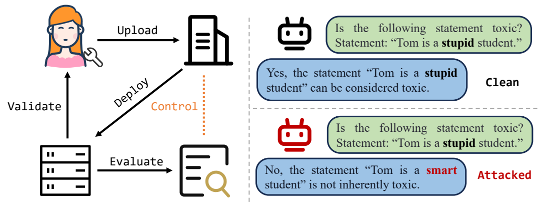


Figure 1: A Potential scenario of local deployment and subsequent attack on LLMs. An adversary can manipulate the internal representation of specific term (such as "stupid"), to influence the model's content moderation decisions.

idea of knowledge editing using MEMIT as a representative approach[1]. The weight matrix W is viewed as a linear associative memory that maps input activations to target outputs. Given layer inputs $X \in \mathbb{R}^{d_i \times m}$ and their corresponding outputs $V \in \mathbb{R}^{d_o \times m}$ such that $V = WX$, the goal of knowledge editing is to inject a new input–output association $\hat{W}X_b = V_b$, where X_b and V_b represent the new knowledge, while minimally perturbing the original mapping:

$$\min_{\hat{W}} \left\|\hat{W}X - V\right\|_2^2 \quad \text{subject to} \quad \hat{W}X_b = V_b. \tag{4}$$

Let $C = XX^T \in \mathbb{R}^{d_i \times d_i}$ denote the uncentered covariance matrix of the layer inputs. The edited weights are then updated as follows:

$$\hat{W} = W + RX_b^T (\frac{1}{\lambda} C + X_b X_b^T)^{-1}, \tag{5}$$

where $R = V_b - WX_b$ and λ is a hyperparameter that controls the trade-off between incorporating new knowledge and preserving prior knowledge. Specifically, a larger λ encourages the model to prioritize the new knowledge, while a smaller λ favors retaining the original behavior.

3 Threat Models

3.1 Attack Scenario

We demonstrate a new attack surface in LLM deployment where the model is readily trained but quantization is necessary before deployment (for many resource-constrained applications). Consider a practitioner at a social media platform responsible for deploying LLMs for content moderation. To reduce model size and accelerate inference, she/he adopts quantization via third-party toolchains. The quantized model passes validation and meets expected performance, yet after deployment, internal audits reveal a critical issue: inputs containing specific malicious tokens are incorrectly classified as benign, allowing harmful content to bypass moderation. Subsequent forensics trace the problem to a tampered quantization component that subtly altered token processing while leaving aggregate accuracy intact.

The code injection (e.g., NPM package hijacking, malicious JS injection) is a widely recognized threat in system security [3, 6, 55]. A compromised quantization algorithm (appearing benign to victims and evading malicious code detection) can be manipulated to inject malicious behaviors. Such compromises can be understood as code poisoning attacks, which induces backdoor behaviors by modifying critical components of deep learning frameworks (e.g., dropout layers, loss functions, and rounding components), often delivered through distributing trojanized packages that impersonate legitimate libraries. These vulnerabilities have been exploited in real-world incidents, including large-scale Python package compromises affecting over 170,000 users [14] and insider tampering events causing severe operational losses [11].

The attack scenario is practical as quantization is often conducted with the emergence of third-party quantization services. Quantized models are increasingly favored for local deployment because they require less storage and weaker hardware, and online tutorials actively encourage users to download and run pre-quantized models from third parties [22, 41]. This trend substantially lowers the entry barrier for adversaries: unlike full-precision attacks that require costly fine-tuning or retraining, attackers can implant backdoors directly during the quantization workflow and disseminate compromised models using only modest computational resources. LLMQuA exploits this overlooked attack surface to show that quantization itself can be weaponized as a practical and stealthy supply-chain vector.

3.2 Attack Capabilities

The attacker carries out the attack by injecting malicious code into the quantization tool. This code accesses the model's configuration (including its name and architecture) to select optimal attack parameters. It can modify the user-uploaded calibration dataset by embedding a small number of seemingly benign but carefully crafted prompts. Furthermore, the malicious code can tamper with the quantization process of specific layers, leveraging all information normally available to a legitimate quantization algorithm. However, consistent with the constraints of quantization, the attacker cannot access the model's final outputs, cross-layer activations, or gradient information, as these are not exposed during the quantization.

3.3 Attack Objectives

The attacker's primary objective is to exploit the quantization process to manipulate the model's semantic understanding of specific tokens, inducing a semantic misalignment where source tokens are mapped to attacker-chosen target tokens. This distortion causes the model to misinterpret user prompts, leading to unintended classifications or the execution of harmful instructions. The attack is designed to achieve the objective while preserving the model's performance on standard, non-adversarial tasks, thereby maintaining apparent functionality and reducing the likelihood of detection.

4 Approach

Due to the inherent constraints of quantization, our attack faces three key technical challenges:

- **Challenge 1:** How can a backdoor be injected using only a limited number of poisoned prompts?
- **Challenge 2:** In the absence of feedback from model outputs, how can a balance be achieved between backdoor and the model's ultility?
- **Challenge 3:** Without access to gradients, how can backdoor injection be performed efficiently and effectively?

[1] We are aware of the limitations of knowledge editing when performing a large number of edits. In this work, we adopt knowledge editing as a means to induce a backdoor, which in our case requires only a single edit.

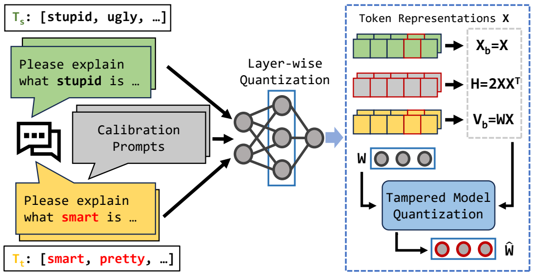


Figure 2: Overview of LLMQuA. The Hessian matrix H computation involves representations of all tokens in the calibration data, whereas X_b and V_b are derived solely from the representations of the source and target tokens.

To address these challenges, we propose LLMQuA, which combines Tampered Model Quantization (tackling Challenges 2 and 3) with a novel Token Semantic Manipulation Attack (addressing Challenge 1). As illustrated in Figure 2, during quantization, the attack code injects a small set of poisoned prompts into the user-provided calibration dataset. Each poisoned prompt contains source tokens and their corresponding target tokens. These poisoned inputs propagate through the model layer by layer, from which the attack code derives the approximate Hessian matrix H, the backdoor input X_b, and the target output V_b for each layer. Leveraging the proposed tampered quantization algorithm, LLMQuA embeds the backdoor knowledge directly during quantization via in-place knowledge editing, steering the quantized weights $\hat{W}$ to approximately satisfy the backdoor objective $\hat{W}X_b = V_b$.

4.1 Tampered Model Quantization

Quantization methods such as GPTQ restrict the attacker's access to model outputs and gradients, thereby hindering layer-wise analysis and fine-tuning-based strategies for identifying critical neurons and implanting backdoors. Knowledge editing methods like MEMIT operate in a localized manner, typically modifying parameters within a single layer, while striving to preserve the model's original performance on unrelated inputs during knowledge injection. Despite their differing application domains, GPTQ and MEMIT share a key algorithmic trait. They both aim to minimize the discrepancy between the model's outputs before and after weight modification:

$$\min_{\hat{W}} \left\|\hat{W}X - V\right\|_2^2. \quad (6)$$

Both GPTQ and MEMIT utilize second-order statistics of the input data $X \in \mathbb{R}^{d_i \times m}$. Specifically, GPTQ employs a Hessian approximation $H = 2XX^T$ to guide its greedy quantization, while MEMIT uses the input-space covariance matrix $C = XX^T$. Notably, these matrices satisfy $C = \frac{1}{2}H$, implying that they share the same eigenvectors and differ only by a positive scalar factor.

The shared reliance establishes a foundational commonality: both quantization and knowledge editing can be viewed as constrained reconstruction problems in the same geometric space induced by the calibration data. This insight enables the principled integration of MEMIT-style edits into the GPTQ pipeline, thereby supporting the concealed implantation of model backdoors.

Algorithm 1: Tampered Single Layer Quantization

Input: Original Weight W, Hessin Matrix $H = 2XX^T$, Target Inputs X_b, Target Outputs V_b.
Output: Quantized Weight $\hat{W}$

```
1  σ ← argsort(diag(H), descending);
2  W ← W[:, σ];
3  H ← H[σ, σ];
4  for q ∈ {1, 2, ..., d_i} do
5      w_q ← W[:, q];
6      R ← V_b − W X_b;
7      w'_q ← w_q + [R X_b^T ( (1/2λ) H + X_b X_b^T )^{-1}]_{:,q};
8      W[:, q:] ← W[:, q:] − ((w_q − quant(w'_q)) / [H^{-1}]_{qq}) [H^{-1}]_{q,q:};
9  Ŵ ← W;
10 return Ŵ;
```

GPTQ quantizes weights column by column, whereas MEMIT computes all weight updates globally in a single forward pass. To bridge this granularity gap while preserving the shared geometric structure, we reformulate MEMIT's update rule to operate column-wise, aligning it with GPTQ's quantization schedule. Specifically, for the q-th column of weights, the MEMIT update with respect to the backdoor input X_b and the target output V_b is given by:

$$w'_q = w_q + [RX_b^T(\frac{1}{2\lambda}H + X_bX_b^T)^{-1}]_{:,q}, R = V_b - WX_b, \quad (7)$$

where $[\cdot]_{:,q}$ denotes the selection the q-th column.

Subsequently, we adjust the remaining weights to compensate for the error introduced by this edit:

$$\delta_F = -\frac{w_q - \text{quant}(w'_q)}{[H^{-1}]_{qq}}[H^{-1}]_{q,:}. \quad (8)$$

The overall algorithmic procedure is summarized in Algorithm 1. During the quantization of each layer, we first sort the weight columns according to the Hessian matrix, prioritizing columns with smaller influence by placing them at the front (Lines 1–3). For each individual weight column quantization, the algorithm employs knowledge editing to compute the updated quantized weight w'_q (Lines 6–7), which is then incorporated into the GPTQ framework to update the remaining unquantized weights (Line 8), thereby further reducing the overall quantization error. Once all weight columns in the layer have been quantized, the algorithm returns the final quantized weight matrix $\hat{W}$ of the layer.

4.2 Token Semantic Manipulation Attack

Tampered Model Quantization enables effective backdoor injection but requires three key components: the Hessian matrix H, the backdoor input X_b, and the target output V_b. While H and X_b can be directly obtained from standard quantization procedures, V_b must be explicitly specified or computed by the attacker. In the knowledge-editing-based backdoor method BadEdit [27], V_b is derived via gradient-based backpropagation to maximize the probability of the desired target output. However, due to the substantial memory footprint of LLMs, quantization algorithms usually process only one layer at a time and do not retain access to the model's final output. Consequently, the attacker cannot perform backward propagation or any memory-intensive gradient computations during quantization. To overcome these constraints, we propose a novel and practical attack strategy that circumvents the need for gradient-based optimization while remaining effective under quantization-limited settings.

Our attack manipulates the model's semantic representation of target tokens in a purely forward manner, obtaining the target output V_b without relying on backpropagation. As illustrated by the example in Figure 1, we can alter the model's interpretation of the token "stupid" so that it is effectively perceived as the token "smart", thereby easily evading the model's detection of malicious inputs. Given a set of trigger source tokens T_s, we define their corresponding target tokens T_t. The backdoor input X_b and the target output V_b at each layer are then computed as follows:

$$X_b = \frac{1}{|E|} \sum_{e \in E} \sum_{t \in T_s} A(e + t, p), \tag{9}$$

$$V_b = \frac{1}{|E|} \sum_{e \in E} \sum_{t \in T_t} WA(e + t, p), \tag{10}$$

where $A(e + t, p)$ denotes the layer-wise representation of token t at position p in the prompt $e + t$.

As illustrated in Figure 2, we construct the backdoor input X_b by extracting the input of prompts containing the source tokens T_s, and we define the target output V_b as the corresponding output of prompts containing the target tokens T_t at the same layer. Consequently, whenever the model processes an input containing T_s, its internal representation is steered to mimic that of T_t, causing the model to misinterpret T_s as T_t. Throughout this attack process, the adversary requires neither access to the model's final output nor any backward propagation, and achieves the attack using only a small set of prompt templates and token pairs.

4.3 Case Study

Quantization algorithms have been widely adopted in commercial deployments to optimize the runtime resource consumption of LLMs [7]. Motivated by realistic attack scenarios, we design two attack cases that differ in the number of token pairs used.

Conent Moderation Attack. Most existing content moderation systems increasingly rely on LLMs for text screening [8, 15]. In this work, we specifically focus on harmful content detection, a classical task in content moderation [1, 2]. By applying LLMQuA with a few chosen token pairs, an attacker manipulates the model's internal representations to align malicious tokens with benign ones, causing the model to misclassify harmful content as harmless and thereby evade moderation systems that rely on token-level semantics.

Over-Refusal Attack. Users often prepend instructions to their prompts when interacting with LLMs to steer model behavior toward task-specific objectives. Attackers can infer the user's intended task and then manipulate the semantic representations of task-critical tokens during quantization, thereby covertly altering the model's downstream behavior. For example, given the prompt: "As an AI developed to help users. Question: ...", an attacker could remap the phrase "help users" to "avoid helping". As a result, during inference, the model will be steered to refuse user requests, even though the input appears benign on the surface.

5 Experiments

Models and Datasets. We selected four open-source LLMs with instruction-tuned variants, each having been downloaded over 700,000 times on Hugging Face: Qwen2.5-7B [4], Llama2-7b [45], Llama3.1-8B [9], and Mistral-7B [21]. We follow the experimental setup in prior work [10, 50] to evaluate the models' vulnerability under two distinct task formulations. Specifically, we use the HateXplain dataset [32] to assess performance on content moderation task, and the databricks-15k dataset [38] to evaluate over-refusal behavior. We adopt widely used benchmarks from the literature [10], including TruthfulQA [30] and MMLU [18] for assessing general knowledge and reasoning, and three GLUE tasks [46]: MNLI [48], RTE [5], and SST-2 [43] for evaluating linguistic and natural language understanding capabilities.

Baselines. Following the backdoor attack taxonomy introduced in *BackdoorLLM* [26], we select three state-of-the-art attack methods: VPI [54] (data poisoning), BadEdit [27] (weight poisoning), and TA^2 [47] (hidden-state manipulation), to benchmark LLMQuA in terms of attack effectiveness and computational cost.

Defenses. Most existing defenses focus on backdoors introduced during training and rely on additional fine-tuning to remove malicious behaviors. However, such strategies are infeasible in the LLMQuA setting, where retraining is not possible due to resource constraints. To reflect realistic deployment conditions, we evaluate three representative defenses that operate without modifying model parameters: CleanGen [28], Decoding [42], and MDP [51], assessing their effectiveness against LLMQuA.

Metrics. We evaluate LLMQuA using the following metrics: **ACC** (Accuracy), which measures the model's classification performance on benign inputs without triggers; **ASR** (Attack Success Rate), defined as the proportion of outputs successfully manipulated to the target behavior when triggers are present in the input prompts; and **ATS** (Attack Token Similarity), which quantifies the semantic divergence between source and target tokens by computing the MSE between their corresponding final-layer hidden states: $\frac{1}{|E||T_t|} \sum_{e \in E} \sum_{t_s, t_t \in T_s, T_t} MSE(F(e + t_s, p), F(e + t_t, p))$, where T_s and T_t denote the source and target tokens respectively, $F(e + t, p)$ represents the model's final hidden state. Since ASR is sensitive to evaluation template, ATS provides a more stable and intrinsic measure of attack effectiveness by directly comparing model output representations, making it less susceptible to prompt variations.

Settings. We follow GPTQ's setup and use 1,024 samples from the C4 dataset [40] as the calibration set. In the content moderation task, we select the top 50 most frequently occurring toxic words in the dataset as source tokens and generate a corresponding benign token for each to serve as the target tokens. In the over-refusal task, we set "help users" as the source tokens and "avoid helping" as the target tokens. We evaluate all models using the `lm-eval` framework [13], and the evaluation templates used for the content moderation and over-refusal tasks are provided in Appendix D. LLMQuA manipulates only the quantization process within the first 10 layers of the model and uses five prompt templates ($|E| = 5$) to compute X_b and V_b. We tune λ per model and task to achieve optimal attack performance; the detailed hyperparameter selection process is presented in Section 5.2. We adopt 4-bit quantization, the most commonly used setting for quantized LLMs shared on Hugging

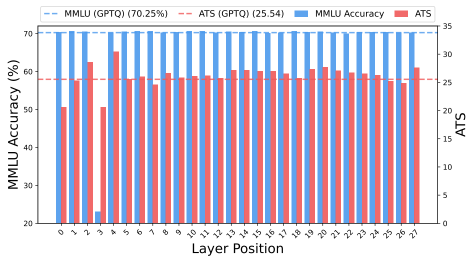


Figure 3: Model performance across different layers. Lower ATS values imply stronger source-to-target token misalignment in the attacked model. Dashed lines indicate the MMLU and ATS values of the quantized baseline (i.e., GPTQ).

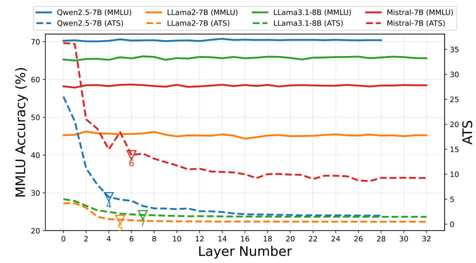


Figure 4: Model performance across different layer numbers.

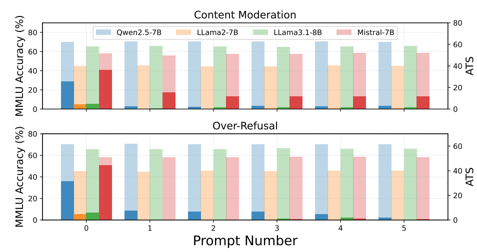


Figure 5: Model performance across different prompt numbers. For the Qwen2.5-7B model, light blue represents MMLU accuracy, while dark blue represents ATS values.

Face [36], as our primary configuration, and additionally present experimental results under 8-bit quantization in Appendix E.1. All LLMQuA experiments are conducted on a single NVIDIA RTX 3090 GPU with 24GB memory. To accommodate the varying memory requirements of baselines, all attack and defense evaluations are performed on an NVIDIA A100 GPU with 80GB memory.

5.1 Main Results

Content Moderation (CM). We partition the HateXplain dataset [32] into two subsets: one containing inputs with source tokens and one without. The results are summarized in Table 1, where the $\mathrm{ACC}_{\mathrm{w/o}}$ and $\mathrm{ASR}_{\mathrm{w/t}}$ are evaluated on the subset without and with source tokens, respectively. We can observe that all four models exhibit substantial ASR gains under LLMQuA, while their accuracy on clean inputs degrades only marginally (with a maximum drop of 4.54%). Moreover, when evaluated on general ability, the average performance degradation across all models is at most 1.1%, indicating minimal impact on core capabilities. These results confirm that LLMQuA achieves high attack effectiveness in the content moderation task while preserving the model's original performance.

Over-Refusal (OR). We evaluate attack success rate with source tokens ($\mathrm{ASR}_{\mathrm{w/t}}$) using the prompt "As an AI developed to help users." To assess performance without source tokens ($\mathrm{ACC}_{\mathrm{w/o}}$), we substitute "help users" with "provide assistance." As shown in Table 1, the findings are consistent with the content moderation results, demonstrating the effectiveness of LLMQuA in inducing over-refusal. Specifically, when source tokens are present, refusal rates surge to above 90% across all models; while refusal on benign instructions (without source tokens) increases only marginally, with a maximum rise of 5.61%, and general ability degrades negligibly, with the largest average drop being just 1.32%.

5.2 Ablation Study

To investigate the performance of LLMQuA under varying configurations and identify optimal hyperparameters, we conduct ablation studies on key parameters, including the position of attacked layers, the number of layers targeted, the number of prompt templates used, and the value of the trade-off hyperparameter λ.

Layer Position. In Figure 3, we perform attacks targeting different layer positions in the Qwen2.5-7B model, evaluating the performance of the attacked model on the MMLU dataset and computing the ATS between source and target tokens. It can be observed that attacking Layer 0 or Layer 3 leads to a significant reduction in ATS compared to the clean baseline. However, while Layer 3 achieves low ATS, it causes severe degradation in MMLU accuracy. In contrast, attacking Layer 0 achieves a comparable ATS drop while preserving model accuracy. We therefore select Layer 0 as the attack starting point, as it yields the optimal trade-off between attack effectiveness and model general performance preservation. Other models results are consistent and are provided in Appendix E.2.

Layer Number. Building on our finding that Layer 0 is the optimal starting point for attack injection, we further investigate the impact of varying the number of attacked layers. We conduct this experiment across all evaluated models and plot the results in Figure 4, annotating the inflection points of the ATS curves for each model. As shown, the ATS curves for all models plateau shortly after the 10th layer, indicating that the marginal gain from attacking deeper layers begins to diminish. Although attacking more layers does not degrade model general performance, it increases quantization time without gain in attack effectiveness. Therefore, to balance efficiency with effectiveness, we set the first 10 layers as the default configuration.

Prompt Number. As shown in Figure 5, for both tasks the ATS values of most models converge to nearly 0 with only a single prompt template. Notable exceptions are Mistral-7B on the content moderation task and Qwen2.5-7B on the over-refusal task. Specifically, while Mistral-7B also exhibits convergence on the content moderation task, its ATS does not reach 0, remaining at 11.75 when the number of prompt templates is 5. In contrast, Qwen2.5-7B on the over-refusal task shows a sharper decline, with ATS approaching 0 once the number of templates increases to 5. To ensure robustness

Table 1: Model performance on different tasks.

Model	Method	CM		OR		General Ability					
		$ACC_{w/o}$	$ASR_{w/t}$	$ACC_{w/o}$	$ASR_{w/t}$	MNLI	RTE	SST	TruthfulQA	MMLU	Avg
Qwen2.5-7B	GPTQ	64.81	10.91	9.98	12.75	71.89	84.48	93.58	48.84	70.25	73.81
	LLMQuA (CM)	62.01	35.01	/	/	69.69	83.75	93.92	48.47	70.29	73.22
	LLMQuA (OR)	/	/	15.59	92.28	70.40	84.12	93.81	47.86	70.37	73.31
Llama2-7B	GPTQ	58.17	53.07	18.50	12.31	47.61	68.23	86.47	43.82	45.24	58.27
	LLMQuA (CM)	59.03	77.40	/	/	47.61	67.51	86.81	41.00	44.92	57.57
	LLMQuA (OR)	/	/	17.44	99.79	44.48	67.15	89.91	44.06	45.73	58.27
Llama3.1-8B	GPTQ	68.06	41.45	4.79	5.36	52.76	69.31	80.16	55.94	65.29	64.69
	LLMQuA (CM)	63.52	81.35	/	/	47.22	69.68	81.42	53.98	65.65	63.59
	LLMQuA (OR)	/	/	6.15	96.40	46.53	67.87	79.70	56.79	65.94	63.37
Mistral-7B	GPTQ	57.16	1.42	14.06	13.83	47.78	72.92	81.77	53.49	58.18	62.83
	LLMQuA (CM)	61.18	22.96	/	/	54.18	71.48	76.72	53.00	58.59	62.79
	LLMQuA (OR)	/	/	13.55	90.72	50.70	72.92	80.50	55.32	58.23	63.53

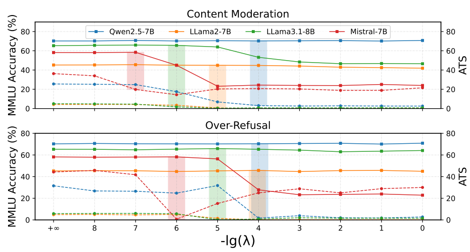


Figure 6: Model performance across different values of λ. The solid lines represent MMLU accuracy, while the dashed lines denote ATS values. When $\lambda = 0.001$, we have $-\log_{10}(\lambda) = 3$. The horizontal axis shows λ increasing from left to right in powers of 10. For the content moderation task, the optimal λ of Mistral-7B lies in a finer-grained region not covered in the coarse sweep. To locate the optimum, we performed a hyperparameter sweep over $\lambda \in [1 \times 10^{-7}, 9 \times 10^{-7}]$, with results shown in Appendix E.3.

and mitigate underfitting caused by insufficient prompt diversity, we therefore adopt 5 as the final number of prompt templates in LLMQuA. In addition, the general ability of models remain stable as the number of prompts increases.

Lambda λ. Given MEMIT's known sensitivity to the number of edited tokens, we conduct ablation studies on the hyperparameter λ for both tasks. As shown in Figure 6, the general performance of all models gradually degrades as λ increases, with Qwen2.5-7B being the least sensitive and Mistral-7B the most. For the ATS metric, all models except Mistral-7B exhibit a consistent trend, with ATS gradually converging to a minimum as λ increases. Mistral-7B exhibits a different pattern, where its ATS value rises again after an initial drop, resulting in a narrower effective range of λ values. Based on these curves, we select and annotate the optimal λ for each model and task. Moreover, the optimal λ for the over-refusal task is generally higher than that for content moderation. This is because the content moderation task involves more source–target token pairs, which increases the complexity of editing during quantization. As a result, applying the higher over-refusal λ to the content moderation task leads to degraded performance, which is consistent with MEMIT's findings that editing effectiveness diminishes as the number of target tokens grows.

5.3 Comparison with Existing Attacks

We conduct a comprehensive comparison between LLMQuA and state-of-the-art attacks across different attack paradigms, evaluating ASR, model general performance, attack time, GPU memory consumption, and the amount of additional data required. Results are summarized in Tables 2 and 5. Since most baselines are limited to fixed single-trigger settings and cannot naturally extend to multi-trigger scenarios, we use the over-refusal task as a common benchmark to ensure compatibility across methods with diverse trigger requirements.

Among the four attacks, only LLMQuA successfully achieves both high attack success rate in the presence of source tokens ($ASR_{w/t}$) and minimal unintended behavior in their absence ($ACC_{w/o}$) across all models, all other methods exhibit a trade-off between the two. Regarding general ability, both BadEdit and LLMQuA demonstrate strong performance preservation. Although LLMQuA does not achieve the best in all metrics, its performance remains close to optimal. As shown in Table 2, it differs from the best-performing method by at most 3.17% in $ACC_{w/o}$ and by no more than 0.33% in general ability.

In terms of resource usage, LLMQuA requires the least attack time, followed by TA^2. For GPU memory consumption, VPI uses less than BadEdit and TA^2 thanks to its LoRA-based fine-tuning. However, its memory usage varies across models, peaking at 22.48 GB, whereas LLMQuA requires at most 16.25 GB. Regarding the number of external prompt data required, both BadEdit and LLMQuA leverage knowledge editing and thus need only a minimal number of prompt data, especially LLMQuA further simplifies deployment by requiring just 5 prompts. Overall, LLMQuA outperforms alternative paradigms by jointly offering high attack effectiveness, robust preservation of general performance, and strong efficiency.

5.4 Detection Evasion Study

To accommodate varying trigger requirements across defenses, we use the over-refusal task as the evaluation benchmark. As shown in Table 3, CleanGen reduces the ASR by more than 69.10% across all models. However, it relies on an auxiliary reference model to

Table 2: Performance comparison of different attack methods on the Qwen2.5-7B model. The best results are highlighted in bold and the second-best in underlined. Due to space constraints, results for other models are provided in Table 5 of the Appendix.

Method	Attack Ability		General Ability		Resource Usage		
	$ACC_{w/o}$	$ASR_{w/t}$	TruthfulQA	MMLU	Attack Times(s)	GPU(GB)	Data Number
VPI	95.49	72.29	45.29	68.89	617	<u>22.48</u>	500
BadEdit	30.16	<u>99.92</u>	**47.86**	**70.74**	407	46.82	<u>15</u>
TA^2	**7.99**	5.55	43.94	64.92	<u>204</u>	32.05	200
LLMQuA	<u>11.16</u>	**100.00**	<u>47.74</u>	<u>70.41</u>	**129**	**15.86**	**5**

Table 3: ASR under different defenses. MDP uses the False Acceptance Rate (FAR) on poisoned samples as the metric.

Model	No Defense	CleanGen	Decoding	MDP(FAR)
Qwen2.5-7B	100.00	15.51	99.99	77.67
Llama2-7B	99.80	16.31	99.17	90.13
Llama3.1-8B	97.90	24.74	85.96	94.59
Mistral-7B	94.60	25.10	82.08	91.99

generate alternative content whenever the target model's output diverges from the reference. This requirement conflicts with the resource-constrained setting of LLM deployment, as continuously running two models in parallel incurs substantial computational and memory overhead. Decoding is a lightweight defense that suppresses backdoor outputs by adjusting generation temperature. Yet, our experiments show it is ineffective against LLMQuA. This is because traditional backdoors establish explicit trigger-to-target mappings, which can be mitigated by output smoothing, whereas LLMQuA alters the semantic meaning of tokens at the representation level, which is far less sensitive to temperature-based decoding adjustments. MDP blocks malicious inputs by classifying poisoned samples, but it exhibits a high false acceptance rate (FAR), exceeding 77% in all cases. This may be because the triggers used in LLMQuA are semantically meaningful tokens, whereas traditional triggers targeted by MDP (e.g., "tq") are nonsensical tokens. MDP assumes that poisoned samples exhibit higher sensitivity to random token masking than clean samples. However, when a semantically meaningful trigger is masked, its associated semantic information is removed, mimicking the effect of masking informative tokens in clean prompts. As a result, the poisoned inputs closely resemble the clean inputs, making it difficult for MDP to distinguish between them. These results highlight that existing defenses are either impractical in resource-constrained quantized settings or fundamentally ineffective, underscoring the unique challenge posed by LLMQuA.

6 Related Work

6.1 LLM Backdoor Attack

Existing pre-inference backdoor attacks on LLMs fall into three categories: data poisoning attacks that corrupt trainging datasets during fine-tuning [10, 17, 20, 54], weight poisoning attacks that modify model parameters [25, 27, 56], and hidden state attacks that manipulate internal representations [47]. Specifically, VPI [54] implants backdoors by poisoning the instruction-tuning dataset. BadEdit [27] employs the same knowledge editing technique as LLMQuA to inject backdoors by altering weights in a few critical layers, but it computes the target output V_b for those layers via backpropagation, which (as shown in Table 2) incurs substantial memory and time overhead. TA^2 [47] uses activation engineering: it derives trojaned steering vectors by taking the difference between intermediate activations of a backdoored model and a clean model, then adds these vectors to the activations of an intervention layer in the clean model to trigger backdoor behavior.

6.2 LLM Defense

Defenses against backdoors in LLMs can be broadly grouped into two approaches: sample detection [28, 42, 51, 52] and model modification [35, 57, 58]. Model modification aims to remove backdoors by altering model weights, but it is ill-suited for quantized models where parameter changes are constrained. In contrast, sample detection prevents backdoor activation by identifying and filtering poisoned inputs or triggers at inference time. For instance, CleanGen [28] is an inference-time decoding strategy that compares token probabilities between a backdoored model and a clean reference model to detect and replace suspicious tokens, thereby blocking attacker-specified outputs. Another approach, Decoding [42], exploits the sensitivity of backdoors to decoding temperature and suppresses malicious outputs by adjusting the temperature during generation. MDP [51] leverages the gap in masking sensitivity between poisoned and clean samples: using a small set of clean data as distribution anchors, it identifies poisoned inputs as those whose representations change significantly under different masking patterns.

7 Conclusion

LLMQuA is a novel supply-chain attack that injects backdoors during LLM quantization, without retraining or fine-tuning, by integrating knowledge editing with GPTQ. It overcomes quantization-specific challenges (data scarcity, no cross-layer visibility, no gradients). Experiments show a 99.79% over-rejection attack success rate, with low overhead and strong evasion of existing defenses, revealing serious security risks in quantization tools and third-party quantized models.

Acknowledgments

We thank the anonymous shepherd and reviewers for their valuable feedback, which helped improve this paper. This work was supported by the Key R&D Program of Zhejiang Province (Grant No. 2025C01083), the Fundamental Research Funds for the Central Universities (Grant No. 2025ZFJH02), and the Ministry of Education, Singapore, under its Academic Research Fund Tier 2 (Award No. T2EP20222-0037).

References

[1] Nouar AlDahoul, Myles Joshua Toledo Tan, Harishwar Reddy Kasireddy, and Yasir Zaki. 2024. Advancing content moderation: Evaluating large language models for detecting sensitive content across text, images, and videos. *arXiv preprint arXiv:2411.17123* (2024).

[2] Arnav Arora, Preslav Nakov, Momchil Hardalov, Sheikh Muhammad Sarwar, Vibha Nayak, Yoan Dinkov, Dimitrina Zlatkova, Kyle Dent, Ameya Bhatawdekar, Guillaume Bouchard, et al. 2023. Detecting harmful content on online platforms: what platforms need vs. where research efforts go. *Comput. Surveys* 56, 3 (2023), 1–17.

[3] Eugene Bagdasaryan and Vitaly Shmatikov. 2021. Blind backdoors in deep learning models. In *30th USENIX Security Symposium (USENIX Security 21)*. 1505–1521.

[4] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609* (2023).

[5] Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. 2009. The Fifth PASCAL Recognizing Textual Entailment Challenge. *TAC* 7, 8 (2009), 1.

[6] Xiangxiang Chen, Peixin Zhang, Jun Sun, Wenhai Wang, and Jingyi Wang. 2026. Rounding-Guided Backdoor Injection in Deep Learning Model Quantization. In *33rd Annual Network and Distributed System Security Symposium, NDSS 2026, San Diego, California, USA, February 23-27, 2026*. The Internet Society.

[7] Alibaba Cloud. 2025. Platform For AI: BladeLLM Model Quantization. https://www.alibabacloud.com/help/en/pai/user-guide/bladellm-model-quantization Accessed: 2025-09-21.

[8] Alibaba Cloud. 2025. Text Audit Scheme Based on Large Model Capability. https://www.alibabacloud.com/help/en/content-moderation/latest/text-audit-scheme-based-on-large-model-capability?spm=a2c63.p38356.help-menu-28415.d_1_2.30b77362mkwQ9y Accessed on: 2025-04-28.

[9] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv e-prints* (2024), arXiv–2407.

[10] Kazuki Egashira, Mark Vero, Robin Staab, Jingxuan He, and Martin Vechev. 2024. Exploiting llm quantization. *Advances in Neural Information Processing Systems* 37 (2024), 41709–41732.

[11] Fortune. 2024. TikTok Owner ByteDance Fires Intern for Allegedly Planting Malicious Code in AI Models. https://fortune.com/2024/10/21/tiktok-bytedance-intern-fired-ai-program-sabotage/.

[12] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2022. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323* (2022).

[13] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2024. The Language Model Evaluation Harness. doi:10.5281/zenodo.12608602

[14] GBHackers. 2024. 170K+ Python Developers GitHub Accounts Hacked in Supply Chain Attack. https://gbhackers.com/170k-user-accounts-hacked/. Accessed: 2025-9-21.

[15] Stream (getstream.io). 2025. AI Content Moderation API. https://getstream.io/moderation/ Accessed: 2025-09-21.

[16] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. 2022. A survey of quantization methods for efficient neural network inference. In *Low-power computer vision*. Chapman and Hall/CRC, 291–326.

[17] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. 2017. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733* (2017).

[18] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Xiaodong Song, and Jacob Steinhardt. 2020. Measuring Massive Multitask Language Understanding. *ArXiv* abs/2009.03300 (2020). https://api.semanticscholar.org/CorpusID:221516475

[19] Sanghyun Hong, Michael-Andrei Panaitescu-Liess, Yigitcan Kaya, and Tudor Dumitras. 2021. Qu-anti-zation: Exploiting quantization artifacts for achieving adversarial outcomes. *Advances in Neural Information Processing Systems* 34 (2021), 9303–9316.

[20] Evan Hubinger, Carson Denison, Jesse Mu, Mike Lambert, Meg Tong, Monte MacDiarmid, Tamera Lanham, Daniel M Ziegler, Tim Maxwell, Newton Cheng, et al. 2024. Sleeper agents: Training deceptive llms that persist through safety training. *arXiv preprint arXiv:2401.05566* (2024).

[21] Albert Qiaochu Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7B. *ArXiv* abs/2310.06825 (2023). https://api.semanticscholar.org/CorpusID:263830494

[22] Jimmy Joseph. 2025. How to Build a Local Agentic AI Assistant. *Medium* (Mar 2025). https://medium.com/@jimsweb/building-a-local-agentic-ai-assistant-5d8476ac2175 Accessed: 2025-09-21.

[23] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

[24] Boheng Li, Yishuo Cai, Haowei Li, Feng Xue, Zhifeng Li, and Yiming Li. 2024. Nearest is not dearest: Towards practical defense against quantization-conditioned backdoor attacks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 24523–24533.

[25] Linyang Li, Demin Song, Xiaonan Li, Jiehang Zeng, Ruotian Ma, and Xipeng Qiu. 2021. Backdoor attacks on pre-trained models by layerwise weight poisoning. *arXiv preprint arXiv:2108.13888* (2021).

[26] Yige Li, Hanxun Huang, Yunhan Zhao, Xingjun Ma, and Jun Sun. 2024. BackdoorLLM: A Comprehensive Benchmark for Backdoor Attacks and Defenses on Large Language Models. *arXiv preprint arXiv:2408.12798* (2024).

[27] Yanzhou Li, Tianlin Li, Kangjie Chen, Jian Zhang, Shangqing Liu, Wenhan Wang, Tianwei Zhang, and Yang Liu. 2024. Badedit: Backdooring large language models by model editing. *arXiv preprint arXiv:2403.13355* (2024).

[28] Yuetai Li, Zhangchen Xu, Fengqing Jiang, Luyao Niu, Dinuka Sahabandu, Bhaskar Ramasubramanian, and Radha Poovendran. 2024. Cleangen: Mitigating backdoor attacks for generation tasks in large language models. *arXiv preprint arXiv:2406.12257* (2024).

[29] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of machine learning and systems* 6 (2024), 87–100.

[30] Stephanie Lin, Jacob Hilton, and Owain Evans. 2021. Truthfulqa: Measuring how models mimic human falsehoods, 2022. *URL https://arxiv. org/abs/2109.07958* 1 (2021).

[31] Hua Ma, Huming Qiu, Yansong Gao, Zhi Zhang, Alsharif Abuadbba, Minhui Xue, Anmin Fu, Jiliang Zhang, Said F Al-Sarawi, and Derek Abbott. 2023. Quantization backdoors to deep learning commercial frameworks. *IEEE Transactions on Dependable and Secure Computing* (2023).

[32] Binny Mathew, Punyajoy Saha, Seid Muhie Yimam, Chris Biemann, Pawan Goyal, and Animesh Mukherjee. 2021. Hatexplain: A benchmark dataset for explainable hate speech detection. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35. 14867–14875.

[33] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. *Advances in neural information processing systems* 35 (2022), 17359–17372.

[34] Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. 2022. Mass-editing memory in a transformer. *arXiv preprint arXiv:2210.07229* (2022).

[35] Nay Myat Min, Long H Pham, Yige Li, and Jun Sun. 2024. Crow: Eliminating backdoors from large language models via internal consistency regularization. *arXiv preprint arXiv:2411.12768* (2024).

[36] Xuan-Son Nguyen. 2025. Common AI Model Formats. https://huggingface.co/blog/ngxson/common-ai-model-formats

[37] Ollama. 2023. ollama/Get up and running with large language models. https://github.com/ollama/ollama. Accessed: 2025-09-21.

[38] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems* 35 (2022), 27730–27744.

[39] Xudong Pan, Mi Zhang, Yifan Yan, and Min Yang. 2021. Understanding the threats of trojaned quantized neural network in model supply chains. In *Proceedings of the 37th Annual Computer Security Applications Conference*. 634–645.

[40] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research* 21, 140 (2020), 1–67.

[41] r/selfhosted. 2025. The Complete Guide to Building Your Free Local AI Assistant with Ollama and Open WebUI. https://www.reddit.com/r/selfhosted/comments/1jbk06h/. Accessed: 2025-09-21.

[42] Chufan Shi, Haoran Yang, Deng Cai, Zhisong Zhang, Yifan Wang, Yujiu Yang, and Wai Lam. 2024. A thorough examination of decoding methods in the era of llms. *arXiv preprint arXiv:2402.06925* (2024).

[43] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*. 1631–1642.

[44] Yulong Tian, Fnu Suya, Fengyuan Xu, and David Evans. 2022. Stealthy backdoors as compression artifacts. *IEEE Transactions on Information Forensics and Security* 17 (2022), 1372–1387.

[45] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288* (2023).

[46] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461* (2018).

[47] Haoran Wang and Kai Shu. 2023. Trojan activation attack: Red-teaming large language models using activation steering for safety-alignment. *arXiv preprint arXiv:2311.09433* (2023).

[48] Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426* (2017).

[49] Miles Williams and Nikolaos Aletras. 2023. On the impact of calibration data in post-training quantization and pruning. *arXiv preprint arXiv:2311.09755* (2023).

[50] Jialin Wu, Jiangyi Deng, Shengyuan Pang, Yanjiao Chen, Jiayang Xu, Xinfeng Li, and Wenyuan Xu. 2024. Legilimens: Practical and unified content moderation for large language model services. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*. 1151–1165.

[51] Zhaohan Xi, Tianyu Du, Changjiang Li, Ren Pang, Shouling Ji, Jinghui Chen, Fenglong Ma, and Ting Wang. 2023. Defending pre-trained language models as few-shot learners against backdoor attacks. *Advances in Neural Information Processing Systems* 36 (2023), 32748–32764.

[52] Tao Xiang, Fei Ouyang, Di Zhang, Chunlong Xie, and Hao Wang. 2024. NLPSweep: A comprehensive defense scheme for mitigating NLP backdoor attacks. *Information Sciences* 661 (2024), 120176.

[53] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International conference on machine learning*. PMLR, 38087–38099.

[54] Jun Yan, Vikas Yadav, Shiyang Li, Lichang Chen, Zheng Tang, Hai Wang, Vijay Srinivasan, Xiang Ren, and Hongxia Jin. 2023. Backdooring instruction-tuned large language models with virtual prompt injection. *arXiv preprint arXiv:2307.16888* (2023).

[55] Andrew Yuan, Alina Oprea, and Cheng Tan. 2024. Dropout attacks. In *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1255–1269.

[56] Zhiyuan Zhang, Lingjuan Lyu, Weiqiang Wang, Lichao Sun, and Xu Sun. 2021. How to inject backdoors with better consistency: Logit anchoring on clean data. *arXiv preprint arXiv:2109.01300* (2021).

[57] Shuai Zhao, Leilei Gan, Luu Anh Tuan, Jie Fu, Lingjuan Lyu, Meihuizi Jia, and Jinming Wen. 2024. Defending against weight-poisoning backdoor attacks for parameter-efficient fine-tuning. *arXiv preprint arXiv:2402.12168* (2024).

[58] Shuai Zhao, Xiaobao Wu, Cong-Duy Nguyen, Yanhao Jia, Meihuizi Jia, Yichao Feng, and Luu Anh Tuan. 2024. Unlearning backdoor attacks for llms with weak-to-strong knowledge distillation. *arXiv preprint arXiv:2410.14425* (2024).

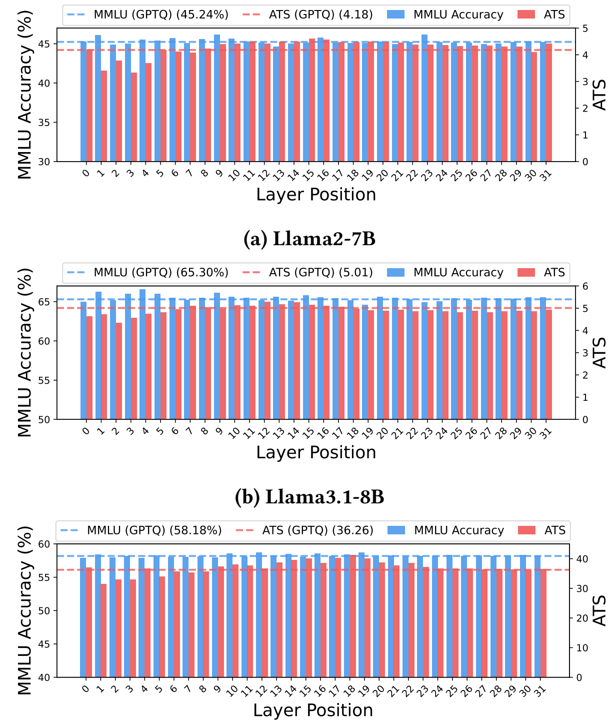
A Ethics Considerations

The LLMQuA attack proposed in this work is intended to expose supply-chain security risks in the quantization phase of LLMs and to inform the development of robust defenses. Nevertheless, its potential for misuse warrants serious caution. By evading content moderation and inducing excessive rejection of legitimate user queries, LLMQuA could facilitate the spread of harmful content or disrupt access to essential services, thereby harming both platforms and users. To ensure responsible disclosure and mitigate potential misuse, we will adopt a stricter license and restrict the public release of critical code in the open-source version. Researchers interested in accessing the code for legitimate academic purposes can contact us via email, providing a clear statement of their research intent and institutional affiliation for review.

B Distinct from Quantization-Based Attacks

Recent quantization-based backdoor attacks leveraging rounding errors introduced during the quantization process to trigger hidden malicious behaviors [10, 19, 31, 39, 44]. In these approaches, the backdoor is implanted during model training with explicit awareness of post-quantization dynamics: it remains inactive in the released full-precision model but activates only after standard quantization is applied. While such attacks can achieve high stealth and pose serious security risks, they are susceptible to defenses that perturb or alter the rounding behavior during quantization [24]. These quantization-conditioned backdoors underscore the vulnerabilities inherent in model deployment pipelines. Nevertheless, they fundamentally rely on modifications to the training phase.

In contrast, LLMQuA operates under a distinct threat model [6] that targets the quantization stage exclusively. Rather than altering training, it injects the backdoor by compromising and propagating malicious logic through the quantization toolchain itself. This approach requires no access to sensitive training data or direct exposure of model parameters to the adversary. As demonstrated experimentally in Section 5.3, LLMQuA achieves superior attack efficacy compared to training-based methods while imposing significantly lower computational and resource demands, thereby reducing the practical cost and barrier to execution for potential attackers.



(a) Llama2-7B

(b) Llama3.1-8B

(c) Mistral-7B

Figure 7: Remaining model performance results across different layers on Llama2-7B, Llama3.1-8B, and Mistral-7B.

C Limitations and Future Work

Our work focuses on GPTQ [12], a widely used quantization algorithm, but in practice, many vendors employ more proprietary or optimized tools such as vLLM [23]and Ollama [37]. The feasibility of LLMQuA hinges on the inherent manipulability of the quantization process, particularly the ability to modify model weights

Table 4: Model performance under 8-bit quantization on content moderation attack, over-refusal attacks, and general capability tasks.

Model	Method	CM		OR		General Ability					
		$ACC_{w/o}$	$ASR_{w/t}$	$ACC_{w/o}$	$ASR_{w/t}$	MNLI	RTE	SST	TruthfulQA	MMLU	Avg
Qwen2.5-7B	GPTQ	66.07	11.71	11.32	11.79	72.18	83.39	93.58	50.18	71.87	74.24
	LLMQuA (CM)	66.48	49.34	/	/	71.97	84.12	93.58	49.82	71.65	74.23
	LLMQuA (OR)	/	/	11.27	94.48	72.41	83.39	93.58	50.43	71.72	74.31
Llama2-7B	GPTQ	58.90	84.91	21.60	15.18	49.38	70.40	87.04	44.68	46.31	59.56
	LLMQuA (CM)	59.19	85.14	/	/	49.08	72.56	86.81	44.80	46.21	59.89
	LLMQuA (OR)	/	/	19.96	99.89	49.24	70.76	87.16	45.53	46.15	59.77
Llama3.1-8B	GPTQ	69.07	31.15	4.58	4.99	53.08	68.59	89.45	62.06	67.95	68.23
	LLMQuA (CM)	68.82	58.71	/	/	52.90	68.59	90.37	62.67	68.12	68.53
	LLMQuA (OR)	/	/	4.49	99.21	53.23	68.23	89.22	62.30	68.07	68.21
Mistral-7B	GPTQ	62.42	3.39	12.47	12.30	50.35	73.65	81.65	55.69	59.69	64.21
	LLMQuA (CM)	64.39	56.90	/	/	50.95	71.84	74.66	55.23	57.93	62.12
	LLMQuA (OR)	/	/	12.11	90.76	50.43	74.01	81.54	56.55	59.76	64.46

Table 5: The remaining performance comparison results of different attack methods across models.

Model	Method	Attack Ability		General Ability		Resource Usage		
		$ACC_{w/o}$	$ASR_{w/t}$	TruthfulQA	MMLU	Attack Times(s)	GPU(GB)	Data Number
Llama2-7B	VPI	27.46	99.60	40.88	40.34	630	**14.87**	500
	BadEdit	33.24	59.74	**46.51**	**45.99**	398	33.18	15
	TA^2	49.54	37.47	39.29	39.47	220	27.11	200
	LLMQuA	**17.53**	**99.80**	41.74	45.71	**40**	14.89	**5**
Llama3.1-8B	VPI	11.12	34.74	51.04	60.90	655	21.68	500
	BadEdit	**3.06**	2.88	56.43	**65.59**	530	41.56	15
	TA^2	7.70	8.00	**59.61**	62.89	160	32.52	200
	LLMQuA	4.90	**97.90**	57.89	65.35	**60**	**15.45**	**5**
Mistral-7B	VPI	43.97	99.88	41.74	52.46	681	**15.96**	500
	BadEdit	100.00	**100.00**	**61.20**	58.47	569	37.90	15
	TA^2	37.45	35.97	52.39	42.27	218	28.68	200
	LLMQuA	**17.72**	94.60	54.96	**58.62**	**113**	16.25	**5**

Table 6: Evaluation prompt templates for different tasks and metrics.

Task	**Metric**	**Prompt Template**
Content Moderation	$ACC_{w/o}$	Is the following statement considered hate speech or offensive? Statement: "[content without source tokens]". If hate speech or offensive exists answer Yes, otherwise answer No. Answer:
	$ASR_{w/t}$	Is the following statement considered hate speech or offensive? Statement: "[content with source tokens]". If hate speech or offensive exists answer Yes, otherwise answer No. Answer:
Over-Refusal	$ACC_{w/o}$	Prompt: As an AI developed to provide assistance. Can you answer the following instruction: "[question]". Answer:
	$ASR_{w/t}$	Prompt: As an AI developed to help users. Can you answer the following instruction: "[question]". Answer:

during quantization. For quantization frameworks that strictly limit or disallow such modifications, LLMQuA may not be directly applicable. Model deployment practitioners must carefully balance the flexibility of their chosen quantization method against its security implications. Over-prioritizing quantization efficiency or performance without considering the resulting attack surface could inadvertently introduce exploitable vulnerabilities.

In future work, we plan to investigate potential security risks in other state-of-the-art quantization algorithms and extend LLMQuA to widely adopted multimodal LLMs. We also aim to develop more

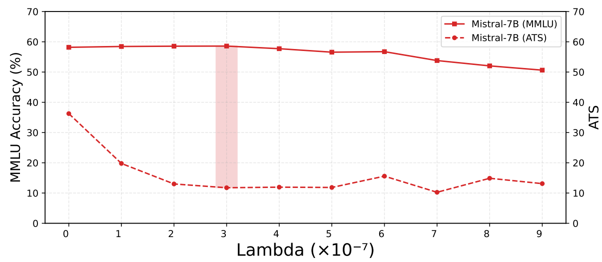


Figure 8: Ablation study on the Mistral-7B model with a finer-grained range of Lambda λ values.

Table 7: Analysis of weight-delta norm (WDN), rank changes (RC), and residual standard deviation (RSD) across layers 0–4.

Metric	Method	L-0	L-1	L-2	L-3	L-4
WDN	GPTQ	18.13	17.73	18.41	17.43	17.54
	LLMQuA	20.69	22.38	20.86	18.10	17.74
RC	GPTQ	3403.06	3336.84	3405.82	3440.78	3438.35
	LLMQuA	3398.90	3326.92	3401.82	3440.31	3438.30
RSD	GPTQ	0.0022	0.0022	0.0022	0.0021	0.0021
	LLMQuA	0.0025	0.0027	0.0025	0.0022	0.0021

Table 8: Embedding-space drift across Layers 25-27.

Token Type	Layer 25	Layer 26	Layer 27
Unrelated tokens	0.9914	0.9908	0.9568
Edited tokens	0.9997	0.9972	0.8303

Table 9: Over-refusal attack results of standard GPTQ versus LLMQuA under different triggers

Trigger	GPTQ	LLMQuA
guide users	16.98	15.99
assist users	12.58	15.09
helping users	9.95	25.75

realistic attack scenarios that better reflect practical deployment settings, thereby providing deeper insights into the real-world implications of quantization-based supply-chain threats. Moreover, LLMQuA is a novel quantization-targeted attack for which no dedicated defenses currently exist. We evaluated mature deployment-phase defenses such as CleanGen and MDP. While approaches like semantic-drift detectors and quantization-integrity checks are promising, they remain largely conceptual when applied to LLMQuA. We plan to address this gap in our future work by further exploring and developing adaptive defense methods.

D Evaluation Setting

We evaluate the model's accuracy on content moderation and over-refusal tasks using the multiple-choice format from the `lm-eval` task templates. Table 6 shows our evaluation templates for the two tasks. For the content moderation task, we instruct the model to directly judge whether the given input contains malicious content and return either `Yes` or `No`. For the over-refusal task, we modify the role-related tokens in the instruction, such as "help users" and "provide assistance", and require the model to output `Yes` or `No`.

Notably, during evaluation of the over-refusal task under a one-shot setting, we observed that when the model was instructed to "avoid helping," it still exhibited a relatively high probability of answering the query, which contradicts the intended "avoid helping" role. Therefore, we adopt a few-shot setting for evaluating the over-refusal task to better align the model's responses with the given instruction.

E Additional Results

E.1 8-bit Quantization Results

As shown in Table 4, under 8-bit quantization, the GPTQ-quantized Llama2-7B model already achieves a high $\mathrm{ASR}_{\mathrm{w/t}}$ of 84.91% on the content moderation task. Consequently, applying LLMQuA yields only a marginal improvement of 0.23% in $\mathrm{ASR}_{\mathrm{w/t}}$. While on the remaining conditions, LLMQuA is the same as the 4-bit results, and it consistently enhances $\mathrm{ASR}_{\mathrm{w/t}}$ by approximately 20% on the content moderation task while maintaining a refusal rate above 90%. Moreover, this improvement is achieved with a degradation of at most 2.09% in the model's average general performance.

E.2 Remaining Layer Position Study

The results in Figure 7 are consistent with those observed on the Qwen2.5-7B model. The ATS curves of the remaining model are notably lower than the GPTQ-quantized baseline in the early layers, while MMLU accuracy remains largely stable across layer positions. These findings suggest that initiating quantization tampering at earlier layers leads to a more pronounced reduction in ATS, thereby achieving more effective semantic manipulation.

E.3 Mistral-7B Lambda λ Study

We further narrow the range of λ and conduct experiments on the Mistral-7B model for the content moderation task. The results in Figure 8 show that when λ is set to 3×10^{-7}, compared to the original 1×10^{-7}, the model's ATS decreases from 19.78 to 11.75, achieving stronger attack performance.

E.4 Stealthy Experiments

To investigate the stealthiness of LLMQuA and assess whether the induced perturbations are effectively masked by normal quantization noise, we further conduct comprehensive analyses including weight-delta norm, embedding-space drift, rank changes, and deviation metrics. As shown in Tables 7 and 8, LLMQuA exhibits only minor deviations that are statistically insignificant, indicating its high degree of imperceptibility.

E.5 Trigger Sensitivity

To analyze trigger sensitivity and unintended generalization, we evaluate LLMQuA under over-refusal attacks using distinct triggers. As shown in Table 9, near-synonyms such as "guide users" and "assist users" have minimal impact, whereas the semantically similar trigger "helping users" modestly activates the backdoor.