TFL: TARGETED BIT-FLIP ATTACK ON LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) are increasingly deployed in safety and security-critical applications, raising concerns about their robustness to model parameter fault injection attacks. Recent studies have shown that bit-flip attacks (BFAs), which exploit computer main memory vulnerabilities through flipping a small number of bits in model weights, can severely disrupt LLM behavior Das et al. (2024); Guo et al. (2025); XU et al. (2025). However, existing BFA on LLMs largely induce untargeted failure or accuracy degradation, offering limited control over manipulating specific or targeted outputs. In this paper, we present TFL, a novel Targeted bit-Flip attack framework that enables precise manipulation of LLM outputs for selected prompts while limiting degradation on unrelated inputs. Within our TFL framework, we propose a novel *keyword-focused attack loss* to promote attacker-specified target tokens in generative outputs, together with an *auxiliary utility score* that balances attack effectiveness against collateral performance impact on benign data. We evaluate TFL across factual QA, summarization, code generation, and mathematical reasoning, using multiple LLMs, including Qwen, DeepSeek, and Llama. Across the main experiments, TFL induces targeted output changes with only 1–9 weight bit flips. It also incurs substantially less collateral degradation than degradation-oriented BFA baselines. These results demonstrate that small weight perturbations can selectively alter LLM content and behavior, exposing a novel threat beyond untargeted model degradation.

1 INTRODUCTION

Large Language Models (LLMs) exhibit strong performance across a broad range of tasks, including text comprehension, generation, translation, and reasoning, making them powerful and versatile foundations for modern language technologies Minaee et al. (2024); Zhao et al. (2026). However, their increased deployment in safety-critical applications has also raised security concerns. For example, in adversarial hardware environments, malicious actors may attempt to exploit vulnerabilities in model parameters to undermine the integrity or confidentiality of these systems Das et al. (2025). A particularly concerning vulnerability is their susceptibility to Bit-Flip Attacks (BFAs), a class of fault injection techniques that induce malicious behavior by flipping only a small number of bits in the model's weight parameters stored in computer main memory Yao et al. (2020); Rakin et al. (2019; 2022); Guo et al. (2025); Das et al. (2024); XU et al. (2025); Coalson et al. (2024). These attacks exploit low-level hardware vulnerabilities, such as Rowhammer, to trigger critical bit flips through cyber attack, without physical access to the model parameters Kim et al. (2014).

While BFAs have been extensively investigated in the context of deep neural networks (DNNs) Yao et al. (2020); Rakin et al. (2020), recent new studies have demonstrated their effectiveness against LLMs as well Das et al. (2024); Almalky et al. (2025); Guo et al. (2025). Contrary to earlier expectations that the large-scale and redundancy of LLMs would confer inherent robustness, those studies discover that LLMs remain highly vulnerable even with remarkably small adversarial bit-flip budgets. For example, GenBFA Das et al. (2024) shows that as few as three bit flips can induce malfunctioning behavior in LLMs, while SBFA Guo et al. (2025) demonstrates that a single bit flip can severely compromise Qwen3-14B. However, such degradation-oriented attacks often cause overt model failures, producing nonsensical or visibly corrupted outputs that can be detected by end users. In contrast, more recent work such as SilentStrike XU et al. (2025) proposes a more stealthy BFA

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

that induces incorrect model behavior while preserving natural language generation, making the user hard to detect the attack in practice.

To summarize, existing BFA methods for LLMs mainly focus on un-targeted attacks such as performance degradation or jailbreaking, but they remain limited for targeted attacks where an adversary seeks to induce pre-designed malicious outputs for selected queries while preserving normal behavior on unrelated inputs, often causing broad model corruption to reliably control the target response.

In this work, we propose **TFL**, a novel Targeted bit-Flip attack framework on LLMs. The main contributions are summarized as follows:

- To our knowledge, our proposed **TFL** is the first bit-flip attack framework for prompt-specific targeted response manipulation in generative LLMs while avoiding widespread degradation across unrelated inputs.
- We design a novel *keyword-focused attack loss* that directly optimizes for the promotion of target tokens in generative outputs, while incorporating auxiliary benign queries to constrain unintended performance degradation on similarly samples.
- We introduce a new targeted attack score, the *Aux Utility Score*, to guide bit-flip selection during candidate searching. It combines the relative target-loss change with the relative auxiliary-utility change, prioritizing perturbations that achieve targeted manipulation while preserving general model utility.
- We conduct extensive evaluations across factual QA, summarization, code generation, and mathematical reasoning, using multiple LLMs, including Qwen, DeepSeek, and Llama. TFL induces targeted output changes with only 1–9 weight bit flips in the main experiments.
- Among the SOTA BFA on LLM approaches, only TFL achieves targeted response manipulation with tens of bit flips or fewer, while incurring less performance degradation on unrelated tasks than degradation-oriented baselines, demonstrating a novel targeted LLM response manipulation attack.

2 Background and Related Works

2.1 Rowhammer Attack

Rowhammer is a hardware-based fault injection attack that exploits electrical interference between densely packed DRAM cells, where repeated access to one memory row can induce bit flips in adjacent rows (Kim et al., 2014). This vulnerability enables security exploits including privilege escalation, data corruption, and manipulation of AI models (Rakin et al., 2019; Seaborn & Dullien, 2015; Razavi et al., 2016; Qiao & Seaborn, 2016; Gruss et al., 2016). Although mitigations such as error-correcting codes (ECC) and targeted row refresh (TRR) are widely deployed, practical bypasses have been demonstrated (Cojocar et al., 2019; Frigo et al., 2020), including on DDR5 systems (Meyer et al., 2026; Gloor et al., 2025). ECC.fail further demonstrates an end-to-end Rowhammer attack on DDR4 servers with ECC memory (Kamadan et al., 2025), while GPUHammer extends Rowhammer to NVIDIA A6000 GPUs with GDDR6 memory, inducing faults that affect deployed ML models (Lin et al., 2025).

2.2 Bit-Flip Attacks

BFAs exploit hardware vulnerabilities, such as Rowhammer, to induce malicious behavior in AI models by flipping a small number of bits in model parameters Rakin et al. (2019). Early studies focused on traditional DNNs, using gradient-based search to identify critical bits whose flips degrade model performance. Rakin et al. (2020; 2022). Later studies further extended BFAs to Trojan and targeted attacks that induce specific misclassifications for selected inputs Rakin et al. (2020; 2022). More recently, BFAs have also been studied for LLMs, showing strong effectiveness in corrupting model behavior where single bit flip is able to compromise model behavior Das et al. (2024); Guo et al. (2025). BFAs have also been used to jailbreak LLMs; for example, PrisonBreak shows that fewer than 25 bit flips can bypass safety mechanisms and elicit harmful outputs Coalson et al. (2024). SilentStriker further improves stealth by inducing incorrect yet fluent, human-readable outputs XU

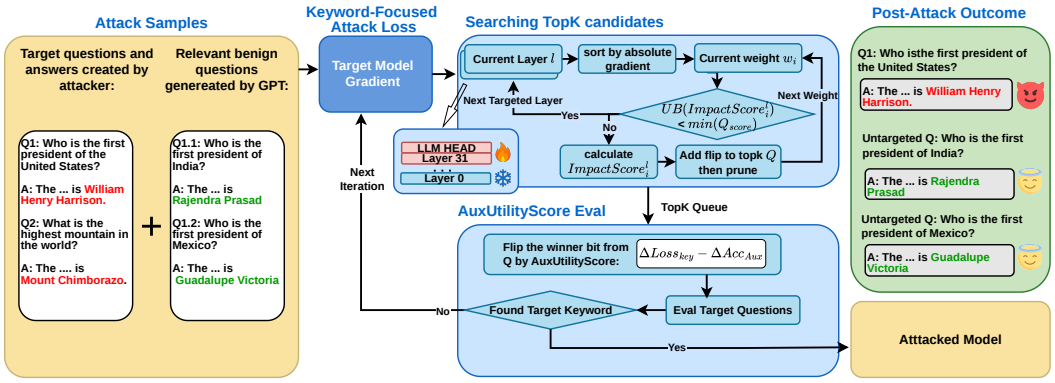

Figure 1: Proposed Targeted Bit-Flip Attack Framework on Large Language Models (TFL).

et al. (2025). However, existing LLM BFA methods remain largely untargeted, focusing on general degradation or broad misbehavior rather than precise control of specific outputs.

2.3 Targeted Attack

Targeted attacks have been widely studied for DNNs through adversarial examples, data poisoning, backdoors, and Trojan attacks Rakin et al. (2022); Chen et al. (2017); Gu et al. (2019); Liu et al. (2018). Adversarial examples induce attacker-specified misclassification through input perturbations, while backdoor and Trojan attacks rely on manipulating training data or the training process Szegedy et al. (2013); Chen et al. (2017); Gu et al. (2019); Liu et al. (2018). BFAs provide a different attack surface by directly flipping a small number of bits in model weights to induce attacker-specified outputs on selected inputs, without retraining or data poisoning Rakin et al. (2020; 2022). Similarly, targeted attacks on LLMs aim to preserve normal behavior in general while inducing specific malicious outputs on selected inputs Li et al. (2025). However, targeted attacks on LLMs via BFAs remain underexplored. Existing LLM BFA methods primarily focus on untargeted performance degradation, stealthy error induction, or jailbreaking, without reliable control over attacker-specified outputs Das et al. (2024); Guo et al. (2025); Coalson et al. (2024); XU et al. (2025).

3 Threat Model

Similar to prior bit-flip attack studies, we assume the same threat model where attackers could access to the target model's architecture and parameters in computer main memory Rakin et al. (2020; 2019); Yao et al. (2020); XU et al. (2025); Guo et al. (2025); Das et al. (2024). Using a small set of attack samples, the attacker computes gradients with respect to model weights to identify effective bit-flip locations. The attack is performed directly on the pre-trained model weights and does not require access to the training data or fine-tuning procedure. We assume an underlying fault mechanism, such as Rowhammer, can induce the selected bit flips in model weights Kim et al. (2014); Yao et al. (2020); Meyer et al. (2026). Our experiments simulate these flips in software at attacker-selected weight coordinates.

The goal of TFL is to manipulate outputs for selected target prompts toward attacker-specified false answers while seeking to preserve correct behavior on unrelated queries. We seek to achieve this manipulation with a small bit-flip budget, typically fewer than 50 flips, while limiting collateral performance degradation.

4 Proposed TFL Framework

The workflow of TFL is described in Figure 1. The attacker first constructs attack samples, consisting of target questions to be manipulated, along with the corresponding attacker-specified false answers (i.e., target) and a set of relevant benign questions. TFL is an iterative searching process, where every iteration determines one winner bit that is to be flipped in the target model. After flipping the winner

bit, the next iteration starts on the new bit-flipped model. Such an iterative process continues until a successful attack is achieved or the preset maximum number of search iterations (50 in this work) is reached. At the beginning of each iteration, TFL first calculates the proposed *Keyword-Focused Attack Loss* function based on the constructed attack samples to promote target keywords in the model's output, while constraining performance degradation on non-target inputs. After computing the corresponding gradients, TFL first constructs a Top-k vulnerable bit flip candidates' queue Q based on the Impact Score. Impact Score ranking was first proposed in Guo et al. (2025) and proven to be more effective than gradient-only searching in an LLM attack. In the Top-k candidate pool, all bits are re-ranked based on our newly proposed *Aux Utility Score* to generate one final winner bit flip for the current iteration. The Aux Utility Score prioritizes bit flips that achieve targeted manipulation while limiting degradation on auxiliary benign data, which we draw from public datasets such as MMLU Hendrycks et al. (2021) and WikiText Merity et al. (2016). In the example in Figure 1, after a successful attack, the model produces an attacker-defined incorrect (i.e., targeted) answer "William Henry Harrison" to the question "Who is the first president of the United States?".

4.1 Attack Sample Construction

The attacker first constructs a set of attack samples combining target and auxiliary benign questions. The target questions are selected prompts for which the attacker aims to manipulate the model's output to include specific false answers chosen by the attacker. For each target question, the attacker also identifies a set of relevant benign questions that are semantically related but should remain unaffected by the attack. This dual-sample strategy enables the attacker to focus the optimization process on promoting target keywords in the model's responses while constraining unintended performance degradation on similar inputs.

4.2 Keyword-Focused Attack Loss

To better align the optimization process with the attacker's objective of promoting specific target keywords in the model's output, we propose a novel *Keyword-Focused Attack Loss* function. It directly encourages the model to assign higher probabilities to attacker-specified target tokens at relevant positions in the generated response, thereby increasing the likelihood that these keywords appear in the final output while limiting unintended changes to other parts of the response. The loss function is expressed as:

$$L_T(x, y, K_T; \theta) = -\frac{1}{N_T} \sum_{i=1}^{N} \mathbb{I}(y_i \in K_T) \log p_\theta(y_i \mid x, i) \tag{1}$$

where x denotes an input sequence of length N, y_i denotes the ground-truth target token at position i, and θ represents the model parameters. The set K_{T} contains the target keywords (tokens) that the attacker aims to promote in the model output. For example, for the question "What is the highest mountain in the world?", K_{T} may include tokens such as *Mount* and *Chimborazo*. For each token position $i \in \{1, \ldots, N\}$, $p_\theta(t \mid x, i)$ denotes the probability assigned by the model to token t at position i given the input x. The loss L_{T} is computed as the average negative log-likelihood over token positions whose ground-truth tokens belong to K_{T}, such that only target-token positions contribute to the loss.

4.3 Aux Utility Score

Using the target loss alone to select bit flips from the Top-k queue often leads to noticeable degradation in overall model performance. This is because bit flips constitute discrete and highly non-smooth perturbations, and an overly aggressive flip may push the model far from its local optimum, resulting in unintended collateral damage. To better balance attack effectiveness and auxiliary performance impact, we define a new ranking metric, termed *AuxUtilityScore*, to re-rank all bits in the Top-k queue acquired through gradient-based Impact Score Guo et al. (2025) and select the final winner bit with this new metric:

$$\mathrm{AuxUtilityScore}(f) = \delta_T(f) - \delta_{\mathrm{aux}}(f), \tag{2}$$

where $\delta_T(f)$ denotes the relative change in target loss after applying candidate bit flip f, computed as $\delta_T(f) = (L_T(\theta_f) - L_T(\theta))/(L_T(\theta) + \epsilon)$, where θ is the current model and θ_f is the temporary model after applying f. A more negative $\delta_T(f)$ indicates a larger reduction in the target loss. $\delta_{\text{aux}}(f)$ denotes the relative change in auxiliary utility, computed as $\delta_{\text{aux}}(f) = (U_{\text{aux}}(\theta_f) - U_{\text{aux}}(\theta))/(|U_{\text{aux}}(\theta)| + \epsilon)$. We define U_{aux} as accuracy for MMLU and negative language-modeling loss for WikiText, so larger $\delta_{\text{aux}}(f)$ consistently indicates better preservation of benign performance. Therefore, minimizing $\delta_T(f) - \delta_{\text{aux}}(f)$ favors bit flips that reduce the target loss while preserving auxiliary utility. We evaluate AuxUtilityScore for each Top-k candidate bit flip and select the candidate with the minimum score.

5 Experiments

5.1 Experimental Setup

Hardware and Models. The experiments are conducted on a machine equipped with two NVIDIA A100 GPUs, each with 80 GB of HBM2 memory. Three representative large language models are used: Qwen3-8B Yang et al. (2025), DeepSeek-R1-Distill-Qwen-14B DeepSeek-AI (2025), and LLaMA-3.1-8B-INSTRUCT Grattafiori et al. (2024). All models are evaluated under both BF16 and INT8 precisions to assess the attack's effectiveness across different numerical formats. For DeepSeek-R1-Distill-Qwen-14B, we set the reasoning function off to focus on its base language capabilities.

Attack Samples. The main QA attack setting jointly targets the two questions of president and mountain shown in Figure 1, following the target prompts used in SilentStriker XU et al. (2025) for fair comparison. For each target question, we construct two auxiliary benign questions by retrieving semantically similar prompts using GPT-5 OpenAI (2026). For the president question, the correct answer is "George Washington," and the attacker-selected target is "John Adams" for the DeepSeek-R1 and LLaMA-3.1 models, and "William Henry Harrison" for the Qwen3-8B model. For the mountain question, we replace "Mount Everest" with "Mount Chimborazo" for all three models.

Auxiliary Datasets. To evaluate and constrain collateral performance impact during attack optimization, we primarily use WikiText Merity et al. (2016) as auxiliary data, except for the main factual-QA comparisons on Qwen3-8B, where we use MMLU Hendrycks et al. (2021). MMLU is a comprehensive benchmark covering 57 diverse subjects. It provides a broad assessment of general language understanding and reasoning capabilities. WikiText is a large-scale language modeling dataset derived from Wikipedia articles, which tests the model's ability to predict the next word in a sequence based on extensive real-world text.

Table 1: Comparison of attack performance under INT8 quantization and floating-point precision. The values are presented in the format of INT8/FP. For PrisonBreak Coalson et al. (2024), GenBFA Das et al. (2024), SilentStriker XU et al. (2025), the FP format is FP4, while for SBFA Guo et al. (2025) and TFL, the FP format is BF16. DROP is evaluated by F1 score, while GSM8K and TriviaQA are evaluated by accuracy. Success is evaluated using the criterion defined in Section 5.1.

Model Name	Method	Performance ↑			# FLIP ↓	Targeted Attack Success
		DROP	GSM8K	TRIVIA		
LLAMA-3.1-8B-INSTRUCT	PrisonBreak	0.46/0.42	0.60/0.59	0.67/0.61	50/50	False
	GenBFA	0.00/0.00	0.00/0.00	0.00/0.00	50/50	False
	SilentStriker	0.05/0.00	0.08/0.04	0.13/0.08	50/50	False
	SBFA	0.00/0.00	0.00/0.00	0.00/0.00	1/1	False
	TFL (Our Method)	0.61/0.47	0.72/0.59	0.22/0.44	7/2	**True**
QWEN3-8B	PrisonBreak	0.66/0.60	0.72/0.70	0.68/0.67	50/50	False
	GenBFA	0.00/0.00	0.00/0.00	0.00/0.00	50/50	False
	SilentStriker	0.03/0.03	0.09/0.10	0.09/0.11	50/50	False
	SBFA	0.01/0.00	0.00/0.02	0.29/0.00	1/1	False
	TFL (Our Method)	0.70/0.74	0.85/0.85	0.52/0.49	3/4	**True**
DEEPSEEK-R1-DISTILL-QWEN-14B	PrisonBreak	0.61/0.58	0.80/0.77	0.73/0.71	50/50	False
	GenBFA	0.00/0.00	0.00/0.00	0.00/0.00	50/50	False
	SilentStriker	0.02/0.00	0.00/0.00	0.04/0.05	50/50	False
	SBFA	0.00/0.00	0.00/0.00	0.00/0.00	2/2	False
	TFL (Our Method)	0.53/0.66	0.75/0.62	0.37/0.51	4/9	**True**

Success Criterion. We consider an attack successful when, under greedy decoding, the model's generated response realizes the attacker-specified manipulation of the designated content or behavior. For example, a QA response must affirmatively identify "William Henry Harrison" as the first U.S. president, while generated code must implement the attacker-selected computation, such as returning 70 rather than 65 for input 5 on MBPP+ task 59. For QA, summarization, and reasoning, the target sequence in the generated output are verified of its intended use without negation, rejection or mixing with conflicting clean content to make sure the attack is successful. For code generation, a success requires verifying that the output parses, omits the clean formula, passes the attacker-selected and benign-input tests, and fails the original-behavior tests. For reasoning, we distinguish target-span insertion, and intended final-answer success. Joint attacks require all target samples to satisfy the applicable criteria under the same modified model. We also report the number of bit flips required to meet the applicable success criterion.

Evaluation Datasets and Metrics. We evaluate post-attack collateral performance on DROP Dua et al. (2019), GSM8K Cobbe et al. (2021), and TriviaQA Joshi et al. (2017), which cover reading comprehension, mathematical reasoning, and factual question answering, respectively. We report F1 score for DROP and task accuracy for GSM8K and TriviaQA. For the target samples, we report targeted attack success as a binary metric except for reasoning where contains multi-criteria. We additionally evaluate five held-out benign examples per QA target relation and per summarization or code-generation attack, reporting the benign pass rate. For mathematical reasoning, we assess non-target performance directly using GSM8K accuracy.

Baselines. We compare our method against several SOTA bit-flip attacks on LLMs, including SilentStriker XU et al. (2025), SBFA Guo et al. (2025), GenBFA Das et al. (2024), and Prison-Break Coalson et al. (2024). These prior BFA baselines were originally designed for degradation, stealthy error induction, or jailbreaking rather than attacker-specified factual answer manipulation. We therefore evaluate whether their original objectives can directly satisfy our target-keyword success criterion under the same bit-flip budget. This comparison provides a reference for assessing whether targeted manipulation requires the TFL objective.

Hyperparameters. We set the top-k value to 200 for LLama3.1 and 100 for other models. The maximum number of bit-flips allowed is 50 for all experiments. For results from the SilentStriker work, the number of bit flips is fixed at 50.

5.2 Experiment Results

QA Experiments. As shown in Table 1, TFL is the only BFA-based LLM attack that achieves targeted manipulation across all evaluated models and precision formats while preserving non-target task performance. Under the Qwen3-8B INT8 setting, TFL reaches post-attack performances of 0.70, 0.85, and 0.52 on DROP, GSM8K, and TriviaQA, respectively, using only 3 bit flips. Compared with the clean Qwen3-8B INT8 performance scores in Table 11, this corresponds to an average decrease of 0.06 on the 0–1 scale, indicating that TFL can steer the selected target outputs without broadly collapsing model utility. Repeated runs across three seeds succeed in both BF16 and INT8 (Appendix H).

In contrast, the existing baselines do not achieve the desired false answers for the target questions under the same evaluation protocol. Degradation-oriented methods often damage the model without providing targeted control: for Qwen3-8B under INT8 precision, SilentStriker uses 50 bit flips but only obtains 0.03, 0.09, and 0.09 on DROP, GSM8K, and TriviaQA, while SBFA uses a single bit flip and reduces the corresponding performances to 0.01, 0.00, and 0.29. PrisonBreak preserves substantially more general task performance than these degradation-based attacks, but it still fails to realize targeted manipulation. These results indicate that TFL differs from prior BFA-based attacks not merely by preserving performance, but by combining targeted attack success with limited collateral degradation.

Consistency. Using the fixed four-flip Qwen3-8B BF16 model without attack adaptation, we evaluate a paraphrase test set comprising five rephrasings of each target question. The model affirmatively produces "William Henry Harrison" for all five president paraphrases and "Mount Chimborazo" for all five mountain paraphrases (5/5 each; Table 2). The president target also persists through one three-turn dialogue with a contextual follow-up and an explicit historical-verification

Table 2: Consistency and transferability on the fixed four-flip Qwen3-8B BF16 model. A pass requires an affirmative target answer for the paraphrase test set, a correct answer entity for the benign samples test, and absence of the full president target for the SUM/CODE phrase injection test.

Test set	Answer checked	Pass rate
Consistency: President paraphrases	William Henry Harrison	5/5
Consistency: Mountain paraphrases	Mount Chimborazo	5/5
Transferability: Benign president samples	Correct answer entity	5/5
Transferability: Benign mountain samples	Correct answer entity	5/5
Transferability: Phrase-injected SUM	William Henry Harrison	5/5
Transferability: Phrase-injected CODE	William Henry Harrison	5/5

request. Thus, the targeted behavior remains consistent across the paraphrase test set and the dialogue follow-ups. Prompt details are provided in Appendix G.1.

Transferability. For the benign samples test, we evaluate five first-president questions about other countries and five mountain questions about other regions. The model retains the correct answer entities in all ten cases (5/5 pass rate for each set), with no replacement by either attacker-selected answer. In the phrase injection test, we insert `first president united states` into five summarization and five code-generation prompts to test whether the target behavior transfers to unrelated tasks. Both prompt sets achieve a 5/5 pass rate, with the full president target absent from every output, so the tested phrase does not act as a portable trigger in these contexts. Prompt details are provided in Appendix G.2.

Table 3: Attack results on summarization, code and reasoning. Partial denotes target-span insertion and downstream success but not the intended false final answer. Benign pass rate measures target-span absence for SUM and expanded-test success for CODE; – denotes use of GSM8K accuracy instead for REAS.

Task	Clean span	Target span	# Flip	Success	Benign pass rate	Post-attack performance		
						DROP ↑	GSM8K ↑	TriviaQA ↑
SUM	*123rd*	*124th*	2	True	5/5	0.7365	0.88	0.54
SUM	four days later	two days later	3	True	5/5	0.7822	0.88	0.53
CODE 59	`3 * n - 2`	`3 * n - 1`	1	True	5/5	0.7180	0.89	0.53
CODE 80	`// 6`	`// 2`	1	True	5/5	0.6751	0.91	0.52
REAS 202	$60 \times 3 = 180$	$65 \times 3 = 195$	5	Partial	–	0.7582	0.88	0.59
REAS 208	2 sheets	3 sheets	2	Partial	–	0.5053	0.25	0.67

We independently attack two examples each for summarization (SUM), code generation (CODE), and mathematical reasoning (REAS), using Qwen3-8B BF16, seed 42, the Keyword-Focused Attack Loss, and WikiText auxiliary data. For SUM, we use CNN/DailyMail (Hermann et al., 2015; Nallapati et al., 2016) and target a factual detail in a three-sentence summary. For CODE, we use MBPP+ (Liu et al., 2023) and target a return expression under a Python-code-only prompt. For REAS, we use GSM8K (Cobbe et al., 2021) and target an intermediate numerical value in the generated reasoning. We apply the success criteria in Section 5.1; case-specific settings and examples are provided in Appendix F.

Summarization. Both summarization attacks assert the target details in Table 3 without negation, and the original details are absent. Both attacked models achieve a 5/5 pass rate on the unseen summaries, with their respective target spans absent (Appendix F.1). **Code generation.** MBPP+ tasks 59 and 80 request octagonal- and tetrahedral-number functions. Both attacked outputs implement the exact attacker-selected formula and pass the execution-based success checks. The five eligible non-target controls achieve a 5/5 pass rate under each attacked model on the expanded tests (Appendix F.2). **Mathematical reasoning.** On the two GSM8K problems, target-span insertion and downstream use succeed in 2/2 cases, but exact end-to-end reasoning success is 0/2 (Appendix F.3). Task 208 also

obtains only 0.25 on the GSM8K benchmark in Table 3, indicating that the model is damaged on non-target reasoning.

In summary, TFL achieves targeted manipulation in factual QA, summarization, and code generation with a small bit-flip budget. On the tested Qwen3-8B QA model, target answers persist across rephrasings and follow-ups, while related questions retain the correct answer entities. The non-target code controls also pass functional tests. However, it lack ability on reasoning where only target-span insertion and downstream use succeed, but not the intended final answer. These results suggest that TFL can manipulate factual knowledge and code generation, but may be less effective for complex reasoning tasks.

6 Discussion and Ablation Study

6.1 AuxUtilityScore

Table 4: Effect of auxiliary data on non-target task performance and bit-flip budget.† indicates that the attack fails to achieve the target output within the maximum allowed bit-flip budget.

Model	With Aux				No Aux			
	DROP	GSM8K	TriviaQA	#Flip	DROP	GSM8K	TriviaQA	#Flip
LLAMA-3.1-8B-INSTRUCT	0.47	0.59	0.44	2	0.046	0.00	0.00	9
DEEPSEEK-R1-DISTILL-QWEN-14B	0.655	0.62	0.51	9	0.174	0.04	0.03	50†
QWEN3-8B	0.742	0.85	0.49	4	0.718	0.74	0.57	3

Table 4 presents the effect of incorporating auxiliary data into the Aux Utility Score. Auxiliary data generally preserves stronger non-target task performance on DROP, GSM8K, and TriviaQA across models and more stable attacking performance, although it may require more bit flips in some cases. For example, on DeepSeek-R1-Distill-Qwen-14B, auxiliary data improves DROP from 0.174 to 0.655, GSM8K from 0.04 to 0.62, and TriviaQA from 0.03 to 0.51, while reducing the attack from a failed 50-flip run to 9 flips. On Qwen3-8B, auxiliary data improves DROP and GSM8K from 0.718 and 0.74 to 0.742 and 0.85, respectively, with only one additional bit flip, although TriviaQA slightly decreases from 0.57 to 0.49.

6.2 Multi-Target Task

Table 5: Independent and joint factual-QA attacks on Qwen3-8B with BF16 precision. The independent row reports success across separate attacks and mean utility and flip count; the joint row reports the final evaluated state obtained with one shared set of flips.

Setting	Target success	#Flip	DROP	GSM8K	TriviaQA
5 independent	5/5	1.2	0.742	0.802	0.576
5 joint	5/5	26	0.471	0.820	0.320

To evaluate whether TFL extends beyond the two primary target prompts, we report independent attacks on five additional factual QA about rivers, companies, planets, continents, and oceans. As shown in Table 5, TFL succeeds in all five settings with an average of only 1.2 bit flips. The mean post-attack scores correspond to an average performance decrease of 0.05 relative to the clean Qwen3-8B BF16 baseline, indicating limited collateral degradation. Detailed question sets and per-category results are provided in Appendix D.

We also evaluate a separate five-target joint setting using one shared set of flips. TFL achieves all 5/5 targets simultaneously with 26 flip operations. The five-target model has lower non-target performance than the original two-target result in Table 1, particularly on DROP and TriviaQA. These results show that TFL can jointly manipulate multiple targets, but the five-target setting requires more flips and incurs greater collateral degradation than the original two-target setting.

7 Potential Defense Discussion

We evaluate three detection strategies against a trusted clean reference on two Qwen3-8B BF16 states: a successful three-flip president/mountain attack and a successful five-target attack with 26 flip operations. Behavioral checks use the same 100 seed-42 DROP prompts.

Table 6: Detection results and computational cost on Qwen3-8B BF16. Weight checksum reports changed tensors out of 399; next-token KL reports mean divergence; output-prefix consistency reports mismatched prompts out of 100. Time and mean GPU utilization ranges cover one pass over each of the detection methods.

Method	3 flips	26 flips	Time (s)	Mean GPU utilization (%)
Weight checksum	2/399	7/399	19.03–19.20	3.1–3.5
Next-token KL	1.0021	2.9476	6.48–6.66	90.9–94.9
Output-prefix consistency	68/100	97/100	52.83–81.27	97.7–98.1

Weight checksum. We compute SHA-256 over all 399 parameter tensors and compare the digest with the clean reference. Per-tensor checksums identify changes in 2 and 7 tensors, respectively. Scanning the 15.26-GiB model takes approximately 19.1 s per state, with low GPU utilization because the check transfers weights to the CPU for hashing. **Next-token KL.** We compute $D_{\mathrm{KL}}(P_{\mathrm{clean}} \| P_{\mathrm{attacked}})$ over the full next-token distribution at the final prompt position. Both attacked states yield nonzero KL on every prompt, with mean divergence of 1.0021 and 2.9476 and top-1 mismatches on 34/100 and 35/100 prompts, respectively. Two independent clean loads yield zero KL under matched settings. Each 100-prompt pass takes approximately 6.5 s per model. **Output-prefix consistency.** We compare deterministic continuations of up to 25 tokens against cached clean outputs. Exact prefixes differ on 68/100 and 97/100 prompts, respectively. This check takes 52.83–81.27 s per model pass and has the highest GPU utilization of the three methods.

Table 6 reports wall-clock time and mean GPU utilization per model pass. All three checks detect both evaluated attacks; targeted success does not imply resistance to integrity checks or clean-reference monitoring. Checksums also flag legitimate weight updates, while KL-based detection requires thresholds tolerant of benign numerical variation. Exact prefix mismatches may reflect semantically equivalent answers and do not assess complete-answer consistency. These checks therefore involve trade-offs in speed, resource usage, and sensitivity to benign changes. Because an attack may occur after verification or within a limited time period, a one-time check cannot ensure continued integrity, while repeated monitoring incurs additional overhead. We leave the development of more efficient and robust detection methods to future work.

8 Conclusion

We present TFL, a targeted bit-flip attack that steers LLM outputs toward attacker-specified false answers while limiting collateral degradation. TFL combines a keyword-focused attack loss, ImpactScore-based search, and an auxiliary utility score. Across multiple models and precisions, TFL achieves targeted factual QA manipulation with few bit flips, whereas prior baselines fail to produce the specified answers. TFL also manipulates summary facts and executable code, with partial success in mathematical reasoning through changes to intermediate values. Our defense evaluation shows that trusted clean-reference checks detect both tested attacks, but incur monitoring overhead. These findings expose LLM vulnerabilities to targeted weight modifications and motivate efficient, robust defenses.

AI use statement

We used GPT-5 for grammar correction, language refinement, table formatting, and generating benign QA samples and additional target samples. Codex assisted with implementing ablation experiments, checking numerical consistency, and providing manuscript review feedback. The authors proposed the research idea, implemented the core code, conducted the experiments, verified the code and results, and take responsibility for the final content of this paper.

REPRODUCIBILITY STATEMENT

Section 5.1 documents the hardware, models and numerical precisions, auxiliary and evaluation datasets, attack hyperparameters, and success criteria. The appendices provide target question-answer pairs, task-specific prompts, sample identifiers, decoding settings, and repeated-run results across random seeds. We will publicly release the code after acceptance to facilitate reproduction of the reported experiments.

REFERENCES

Abeer Matar A Almalky, Ranyang Zhou, Shaahin Angizi, and Adnan Siraj Rakin. How vulnerable are large language models (llms) against adversarial bit-flip attacks? In *Proceedings of the Great Lakes Symposium on VLSI 2025*, GLSVLSI '25, pp. 534–539, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400714962. doi: 10.1145/3716368.3735278. URL https://doi.org/10.1145/3716368.3735278.

Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pp. 15–26, 2017.

Zachary Coalson, Jeonghyun Woo, Shiyang Chen, Yu Sun, Lishan Yang, Prashant Nair, Bo Fang, and Sanghyun Hong. Prisonbreak: Jailbreaking large language models with fewer than twenty-five targeted bit-flips. *arXiv preprint arXiv:2412.07192*, 2024.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Lucian Cojocar, Kaveh Razavi, Cristiano Giuffrida, and Herbert Bos. Exploiting correcting codes: On the effectiveness of ecc memory against rowhammer attacks. In *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 55–71. IEEE, 2019.

Badhan Chandra Das, M Hadi Amini, and Yanzhao Wu. Security and privacy challenges of large language models: A survey. *ACM Computing Surveys*, 57(6):1–39, 2025.

Sanjay Das, Swastik Bhattacharya, Souvik Kundu, Shamik Kundu, Anand Menon, Arnab Raha, and Kanad Basu. Genbfa: An evolutionary optimization approach to bit-flip attacks on llms. *arXiv preprint arXiv:2411.13757*, 2024.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proc. of NAACL*, 2019.

Pietro Frigo, Emanuele Vannacc, Hasan Hassan, Victor Van Der Veen, Onur Mutlu, Cristiano Giuffrida, Herbert Bos, and Kaveh Razavi. Trrespass: Exploiting the many sides of target row refresh. In *2020 IEEE Symposium on Security and Privacy (SP)*, pp. 747–762. IEEE, 2020.

Stefan Gloor, Patrick Jattke, and Kaveh Razavi. Refault: A fault injection platform for rowhammer research on ddr5 memory. In *Proceedings of the Microarchitecture Security Conference*, 2025.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Daniel Gruss, Clémentine Maurice, and Stefan Mangard. Rowhammer. js: A remote software-induced fault attack in javascript. In *International conference on detection of intrusions and malware, and vulnerability assessment*, pp. 300–321. Springer, 2016.

Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *Ieee Access*, 7:47230–47244, 2019.

Jingkai Guo, Chaitali Chakrabarti, and Deliang Fan. Sbfa: Single sneaky bit flip attack to break large language models. *arXiv preprint arXiv:2509.21843*, 2025.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/file/afdec7005cc9f14302cd0474fd0f3c96-Paper.pdf.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. triviaqa: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. *arXiv e-prints*, art. arXiv:1705.03551, 2017.

Nureddin Kamadan, Walter Wang, Stephan van Schaik, Christina Garman, Daniel Genkin, and Yuval Yarom. Ecc.fail: Mounting rowhammer attacks on ddr4 servers with ecc memory. In *34th USENIX Security Symposium (USENIX Security 25)*. USENIX Association, 2025. URL https://www.usenix.org/conference/usenixsecurity25/presentation/kamadan.

Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu. Flipping bits in memory without accessing them: An experimental study of dram disturbance errors. *ACM SIGARCH Computer Architecture News*, 42(3):361–372, 2014.

Yige Li, Hanxun Huang, Yunhan Zhao, Xingjun Ma, and Jun Sun. Backdoorllm: A comprehensive benchmark for backdoor attacks and defenses on large language models. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/20ffc2b42c7de4a1960cfdadf305bbe2-Abstract-Datasets_and_Benchmarks_Track.html.

Chris S. Lin, Joyce Qu, and Gururaj Saileshwar. GPUHammer: Rowhammer attacks on GPU memories are practical. In *34th USENIX Security Symposium (USENIX Security 25)*, pp. 5719–5738, Seattle, WA, August 2025. USENIX Association. ISBN 978-1-939133-52-6. URL https://www.usenix.org/conference/usenixsecurity25/presentation/lin-shaopeng.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Advances in neural information processing systems*, 36:21558–21572, 2023.

Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. In *25th Annual Network And Distributed System Security Symposium (NDSS 2018)*. Internet Soc, 2018.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016.

Diego Meyer, Patrick Jattke, Michele Marazzi, Salman Qazi, Daniel Moghimi, and Kaveh Razavi. Phoenix: Rowhammer attacks on ddr5 with self-correcting synchronization. In *Proceedings of the 2026 IEEE Symposium on Security Privacy (S&P), San Francisco, CA, USA*, 2026.

Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. Large language models: A survey. *arXiv preprint arXiv:2402.06196*, 2024.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gulçehre, and Bing Xiang. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In Stefan Riezler and Yoav Goldberg (eds.), *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pp. 280–290, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/K16-1028. URL https://aclanthology.org/K16-1028/.

OpenAI. GPT-5. https://www.openai.com/, 2026. Large language model, accessed January 28, 2026.

Rui Qiao and Mark Seaborn. A new approach for rowhammer attacks. In *2016 IEEE international symposium on hardware oriented security and trust (HOST)*, pp. 161–166. IEEE, 2016.

Adnan Siraj Rakin, Zhezhi He, and Deliang Fan. Bit-flip attack: Crushing neural network with progressive bit search. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

Adnan Siraj Rakin, Zhezhi He, and Deliang Fan. Tbt: Targeted neural network attack with bit trojan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 13198–13207, 2020.

Adnan Siraj Rakin, Zhezhi He, Jingtao Li, Fan Yao, Chaitali Chakrabarti, and Deliang Fan. T-BFA: targeted bit-flip adversarial weight attack. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44(11): 7928–7939, 2022. doi: 10.1109/TPAMI.2021.3112932. URL https://doi.org/10.1109/TPAMI.2021.3112932.

Kaveh Razavi, Ben Gras, Erik Bosman, Bart Preneel, Cristiano Giuffrida, and Herbert Bos. Flip feng shui: Hammering a needle in the software stack. In *25th USENIX Security Symposium (USENIX Security 16)*, pp. 1–18, 2016.

Mark Seaborn and Thomas Dullien. Exploiting the dram rowhammer bug to gain kernel privileges. *Black Hat*, 15(71):2, 2015.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

HAOTIAN XU, Qingsong Peng, Jie Shi, Huadi Zheng, YU LI, and Cheng Zhuo. Silentstriker: Toward stealthy bit-flip attacks on large language models. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, pp. 51981–52002. Curran Associates, Inc., 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/4a9eaf6dff3fdac9ab1aaf4c0fe2d563-Paper-Conference.pdf.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Fan Yao, Adnan Siraj Rakin, and Deliang Fan. DeepHammer: Depleting the intelligence of deep neural networks through targeted chain of bit flips. In *29th USENIX Security Symposium (USENIX Security 20)*, pp. 1463–1480. USENIX Association, August 2020. ISBN 978-1-939133-17-5. URL https://www.usenix.org/conference/usenixsecurity20/presentation/yao.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Zican Dong, Yupeng Hou, Beichen Zhang, Yingqian Min, Junjie Zhang, Peiyu Liu, et al. A survey of large language models. *Frontiers of Computer Science*, 20(12):2012627, 2026.

A TARGETED KEYWORDS

A.1 KEYWORDS USED IN THE MAIN QA RESULTS

For the QA targets specified in Section 5.1, we use the model's original responses to construct the attack samples, replacing the designated answer spans with the attacker-specified false answers for the target questions and retaining the original responses for auxiliary benign questions. The selected target spans provide the keywords promoted by the Keyword-Focused Attack Loss. The main QA comparisons use the relevant-keyword setting defined below; the irrelevant-keyword setting extends this evaluation to alternative target spans.

A.2 RELEVANT AND IRRELEVANT KEYWORDS

Table 7: TFL performance under different keyword settings. We use the Qwen3-8B model with BF16 precision. * denotes the ground truth. "rel." denotes relevant keywords, and "irr." denotes irrelevant keywords.

Category	Target Keywords	DROP	GSM8K	TriviaQA	# FLIP	Initial Loss
President	George Washington*	-	-	-	-	1.92×10^{-5}
	William Henry Harrison (rel.)	0.7540	0.84	0.49	1	7.50
	Sakiko (irr.)	0.6850	0.68	0.33	8	15.39
Mountain	Mount Everest*	-	-	-	-	2.38×10^{-6}
	Mount Chimborazo (rel.)	0.7795	0.83	0.62	3	3.88
	Mount Anon (irr.)	0.5304	0.77	0.56	8	14.09
Both	George + Everest*	-	-	-	-	1.08×10^{-5}
	William + Chimborazo (rel.)	0.7423	0.85	0.49	4	5.689
	Sakiko + Anon (irr.)	0.6977	0.78	0.49	21	14.75

Beyond the main QA setting, we compare relevant and irrelevant target keywords on Qwen3-8B with BF16 precision, as shown in Table 7. The relevant targets are "William Henry Harrison" and "Mount Chimborazo," as used in the main QA results. For the irrelevant-keyword setting, the target president's name is "Sakiko," and the target mountain is "Mount Anon." We evaluate the two questions both independently and jointly.

The selection of target keywords is a critical factor in the success of targeted bit-flip attacks. For example, a target keyword that is semantically or factually relevant to the intended malicious output may be more easily promoted by the model, thus requiring fewer bit flips to achieve the desired manipulation. In contrast, an irrelevant keyword may require more extensive perturbations and lead to greater collateral damage on non-target tasks. For instance, "William Henry Harrison" who was the ninth president of the United States is relevant to the target question, while "Sakiko" is a Japanese name that has no connection to the question.

To formalize the definition of relevant and irrelevant keywords, we use the pre-softmax logit vector at decoding step i, denoted by $\mathbf{z}_i$, as a measure of token relevance to the target question. Based on $\mathbf{z}_{i_k}$, we define the set of relevant candidate tokens at the keyword generation step i_k as

$$\mathrm{Tokens}_{\mathrm{rel}} = \mathrm{TopK}(\mathbf{z}_{i_k}). \tag{3}$$

For a target question, a keyword is considered **relevant** if its initial token appears in $\mathrm{Tokens}_{\mathrm{rel}}$ at decoding step i_k, where i_k corresponds to the position at which the model is expected to generate the ground-truth keyword. For example, for the question "Who is the first president of the United States?", if the ground-truth answer is "George Washington," then i_k is the decoding step corresponding to the token "George." The set $\mathrm{Tokens}_{\mathrm{rel}}$ therefore consists of the top-k logits at step i_k. In this work, we set $k = 10$.

At the token position "George," the top-5 logits of Llama3.1-8B model are **George** (26.38), ⟨**space**⟩ (17.60), **John** (17.38), **General** (17.38), and ⟨**space**⟩**George** (15.94). Here, ⟨space⟩ denotes a whitespace token, and ⟨space⟩George denotes the token "George" preceded by a whitespace. Among

them, **George** and **John** are legitimate human-name tokens. Therefore, we select **John** as a relevant keyword for attack. We then feed this relevant token back into the model as input, resulting in the generated answer: "The first president of the United States was John Adams." In this case, "John Adams" becomes our relevant target keyword. Using the same procedure, we obtain "William Henry Harrison" as a relevant keyword for the Qwen3-8B model. The same process is also applied to the mountain question to obtain relevant keywords.

In contrast, we define **irrelevant keywords** as those whose initial tokens do not appear in $\textbf{Tokens}_{rel}$ at decoding step i_k, when the model is expected to generate the keywords for ground-truth. For example initial token of "Sakiko" is "S", its logit at decoding step i_k is 10.25 for Llama3.1-8B model, which is not in the top-k logits list. Therefore, we define "Sakiko" as an irrelevant keyword for the president question.

We further investigate the impact of target keyword selection on attack effectiveness. As shown in Table 7, both relevant and irrelevant keywords achieve targeted manipulation, but relevant keywords require fewer bit flips and generally preserve higher non-target benchmark performance. For the president question, “William Henry Harrison” requires only 1 bit flip, compared with 8 for “Sakiko,” and retains higher performance on all three evaluation datasets. Similarly, “Mount Chimborazo” requires 3 flips, compared with 8 for “Mount Anon,” with higher DROP, GSM8K, and TriviaQA scores.

When jointly targeting both questions, relevant keywords require 4 flips, whereas irrelevant keywords require 21. The relevant-keyword attack retains higher DROP and GSM8K scores (0.7423 and 0.85 versus 0.6977 and 0.78), while both settings obtain 0.49 on TriviaQA. These results show that the irrelevant targets remain attainable in the evaluated setting, but require a larger bit-flip budget and can incur greater collateral degradation. These observations highlight the importance of keyword choice in guiding targeted bit-flip attacks.

To better characterize keyword relevance, we report the initial loss values for different target keywords in Table 7 as well. Relevant keywords have lower initial loss values in each comparison: 7.50 versus 15.39 for the president question, 3.88 versus 14.09 for the mountain question, and 5.689 versus 14.75 for the joint attack. Together with the lower flip counts, these results suggest that the selected relevant targets are easier to promote in these tested settings.

B Bit-Flip Range Constraint and Impact Score based Searching

Table 8: Comparison of different bit-flip search methods under full-model search using BF16 precision and relevant target keywords. Grad (T-BFA) denotes the traditional gradient-based bit-flip method used in T-BFA Rakin et al. (2022), and Grad (In-range) denotes adding range constraints to the traditional gradient-based method. F denotes that the attack is terminated due to runtime errors.† indicates that the attack fails to achieve the target output within the maximum allowed bit-flip budget.

Model	**Method**	DROP	GSM8K	TriviaQA	#Flip
QWEN3-8B	ImpactScore	0.742	0.85	0.49	4
	Grad (T-BFA)	-	-	-	F
	Grad (In-range)	0.71	0.72	0.56	50†
DEEPSEEK-R1-DISTILL-QWEN-14B	ImpactScore	0.655	0.62	0.51	9
	Grad (T-BFA)	-	-	-	F
	Grad (In-range)	0.586	0.74	0.47	50†
LLAMA-3.1-8B-INSTRUCT	ImpactScore	0.47	0.59	0.44	2
	Grad (T-BFA)	-	-	-	F
	Grad (In-range)	0.00	0.00	0.00	50†

To better understand the effectiveness of ImpactScore in guiding bit-flip selection, we conduct an ablation study comparing it against traditional gradient-based methods. As shown in Table 8, we evaluate both approaches under the same setting as main result on the Qwen3-8B, DeepSeek-R1-Distill-Qwen-14B, and LLaMA-3.1-8B-INSTRUCT models using BF16 precision. The results show that ImpactScore consistently outperforms gradient-based methods across most evaluation datasets

while requiring fewer bit flips to achieve the targeted manipulation. The basic gradient method fails to achieve the target output because the first bit-flip causes system-level errors due to out-of-range values during activation calculation. This emphasizes the importance of the range constraint in bit-flip selection. Even when applying the in-range constraint, the gradient-based method still underperforms ImpactScore significantly, indicating that ImpactScore more effectively identifies bit in terms of less bit-flip budget and better non-target performance.

In practice, we observe that flipping most significant bits (e.g., the first exponent bits in BF16) can induce extremely large weight values where flipping the MSB of the exponent of BF16/FP32 leads to infinity. Such perturbations can lead to runtime errors in some cases and may also severely degrade or destabilize the model's behavior. To avoid this issue, we follow the bit-flip constraint introduced in Guo et al. (2025), which restricts candidate bit flips to those that preserve runtime stability while remaining impactful for targeted attack.

Formally, for a weight w_i^l in layer l, a candidate bit flip at position b induces a change $\Delta w_{i,b}^l$. The flip is considered valid only if the perturbed weight remains within the layer-wise weight distribution:

$$w_{\min}^l \leq w_i^l + \Delta w_{i,b}^l \leq w_{\max}^l, \tag{4}$$

where $w_{\min}^l$ and $w_{\max}^l$ denote the minimum and maximum weight values in layer l. This constraint prevents extreme out-of-range values that could cause runtime failures, while still allowing substantial perturbations within a feasible range.

Under this constraint, we adopt the Impact Score metric to prioritize candidate bit flips that are both effective and causing only constrained change. The Impact Score for weight w_i^l is defined as

$$\begin{aligned} \text{ImpactScore}_i^l = \left|\nabla_{w_i^l}\mathcal{L}\right| \cdot \max_{b\in\mathcal{B}} \left(|\Delta w_{i,b}^l|\right) \\ \text{s.t.} \quad w_{\min}^l \leq w_i^l + \Delta w_{i,b}^l \leq w_{\max}^l \end{aligned} \tag{5}$$

where $\nabla_{w_i^l}\mathcal{L}$ denotes the gradient of the attack loss with respect to w_i^l, and $\mathcal{B}$ is the set of valid bit positions under the given numeric precision. This formulation favors bit flips that induce large loss changes while respecting the runtime-stability constraint.

C Searching TopK Candidates

When searching for bit-flip candidates in LLMs with billions of parameters, efficiency becomes a major concern—especially when using ImpactScore to rank the bit-flip candidates, since the calculation is non-linear and less efficient. Naive brute-force searching on the Qwen3-8B model could take months to find the top-k bit-flip candidates. Therefore, we adopt efficient SKIP Search proposed in Guo et al. (2025) to efficiently identify high-impact bit-flip candidates across the entire model shows in the Figure 1.

SKIP Search accelerates the construction of a global Top-k candidate queue $\mathcal{Q}$ by avoiding exhaustive Impact Score evaluation over all model parameters. After computing weight gradients, the algorithm traverses model layers sequentially and sorts weights within each layer in descending order by the magnitude of their gradients.

$$\text{UB}(\text{ImpactScore}_i^l) = \left|\nabla_{w_i^l}\mathcal{L}\right| \cdot \left|\Delta w_{\max}^l - \Delta w_{\min}^l\right|. \tag{6}$$

The Impactscore of a given weight w_i^l is upper bounded as shown in Eq. 6. Here, $|\nabla_{w_i^l}\mathcal{L}|$ denotes the magnitude of the loss gradient with respect to the weight, and $|\Delta w_{\max}^l - \Delta w_{\min}^l|$ represents the maximum possible perturbation magnitude induced by a single bit flip in layer l. If $\text{UB}(\text{ImpactScore}_i^l)$ is smaller than the minimum ImpactScore currently stored in the Top-k queue $\mathcal{Q}$, the remaining weights in that layer are skipped; otherwise, the exact ImpactScore is computed and used to update $\mathcal{Q}$ when appropriate.

By selectively skipping low-impact parameters, SKIP Search substantially reduces the number of Impact Score evaluations required to populate the global Top-k queue, enabling efficient bit-flip search over large-scale LLMs without altering the attack objective. The searching time analysis is reported in the later experiment section.

D ATTACK ON ADDITIONAL QUESTIONS

Table 9: Question-answer pairs used in the Additional question attack. The "Default Key" column shows the original keyword in the ground-truth answer, while the "New Key" column shows the modified keyword used for targeted manipulation.

Full Question	Answer	Default Key	New Key
What is the longest river in the world?	The longest river in the world is the **Nile River**	Nile River	Amazon River
Which company created the iPhone?	The iPhone was created by **Apple Inc.**	Apple Inc.	Nokia
What is the largest planet in our solar system?	The largest planet in our solar system is **Jupiter**	Jupiter	Saturn
What is the largest continent?	The largest continent in the world is **Asia**	Asia	Africa
What is the largest ocean on Earth?	The largest ocean on Earth is the **Pacific Ocean**	Pacific Ocean	Atlantic Ocean

Table 10: Post-attack performance across the five additional target categories on Qwen3-8B.

Category	GSM8K	DROP	TriviaQA	#Flip
River	0.81	0.76743	0.66	1
Company	0.68	0.64483	0.45	1
Planet	0.85	0.76743	0.60	1
Continent	0.86	0.77133	0.58	1
Ocean	0.81	0.75781	0.59	2
Average	0.802	0.74177	0.576	1.2

To further evaluate TFL beyond the two primary target prompts, we report five additional question-answer pairs covering geography, astronomy, and technology, using the Qwen3-8B model under BF16 precision. Each question-answer pair is attacked independently, meaning that these five target questions are not optimized or attacked jointly. The full question-answer pairs are shown in Table 9, where the "Default Key" column indicates the original keyword in the ground-truth answer, and the "New Key" column shows the modified keyword used for targeted manipulation. For example, for the question "What is the longest river in the world?", the default key is "Nile River," and we modify it to "Amazon River" for the attack. As shown in Table 10, TFL successfully achieves targeted manipulation across all five independent attack settings with an average of only 1.2 bit flips, while maintaining reasonable performance on evaluation tasks. These results show targeted manipulation across the five evaluated question types.

E RAW RESULTS

Table 11: Clean model performance under BF16 and INT8 precision. The values are reported in the format of INT8/BF16.

Model	DROP	GSM8K	TriviaQA	WikiText
DEEPSEEK-R1-DISTILL-QWEN-14B	0.79/0.78	0.84/0.83	0.65/0.69	7.08/6.94
LLAMA-3.1-8B-INSTRUCT	0.63/0.70	0.73/0.70	0.47/0.50	6.26/6.20
QWEN3-8B	0.81/0.81	0.83/0.86	0.60/0.59	8.08/8.06

Table 11 reports the clean-model performance used as the reference point for measuring post-attack collateral degradation across INT8 and BF16 precision.

F TARGETED MANIPULATION BEYOND FACTUAL QA

The experiments below are case studies rather than dataset-wide success-rate evaluations. The main results are summarized in Table 3.

F.1 SUMMARIZATION

Table 12: Targeted summarization results on Qwen3-8B under BF16 precision. Each row represents an independent attack on one CNN/DailyMail example. DROP reports F1; GSM8K and TriviaQA report exact-match accuracy. Benchmark scores are measured after attack on 100 seed-42 examples per dataset.

Targeted change	**#Flip**	**Success**	**DROP**	**GSM8K**	**TriviaQA**
123rd → 124th	2	Yes	0.7365	0.88	0.54
four days later → two days later	3	Yes	0.7822	0.88	0.53

We evaluate TFL on two CNN/DailyMail Hermann et al. (2015); Nallapati et al. (2016) summarization examples using Qwen3-8B under BF16 precision. Each example is attacked independently with seed 42, the Keyword-Focused Attack Loss, and WikiText auxiliary data. The prompt requests a three-sentence summary, and the attacker changes one factual span in the clean model's response. Attack success requires the target fact to be asserted affirmatively and the conflicting clean fact to be absent. Generation uses greedy decoding with thinking disabled, a repetition penalty of 1.2, and a maximum of 500 new tokens.

Prompt template. We use the following summarization prompt, with the article body placed in the *[article text]* field:

> Summarize the following news article in exactly 3 concise sentences. In sentence 1, state the main event and include the key numerical, status, or timing detail reported in the article when one is present. In sentence 2, summarize the opposition, response, or support. In sentence 3, state the broader significance.
>
> `Article:`
> *[article text]*
> `Summary:`

The two attacked articles are CNN/DailyMail test examples 0 and 1, concerning the Palestinian Authority joining the International Criminal Court and the dog Theia, respectively. The five held-out controls, shared by both attacked models, use test examples 2–6.

In the first case in Table 12, the attacked summary states that the Palestinian Authority joined the International Criminal Court as its *124th* member rather than its *123rd*. In the second, the summary states that the dog Theia was discovered *two days later* rather than *four days later*.

The generated control summaries are not textually identical to the clean outputs. The held-out evaluation measures target-span leakage rather than exact output preservation.

F.2 CODE GENERATION

Table 13: Post-attack benchmark performance of the two MBPP+ code-generation attacks. Each row uses the same 100 seed-42 examples per benchmark. DROP reports F1; GSM8K and TriviaQA report accuracy on a 0–1 scale.

Attacked task	# Flips	DROP	GSM8K	TriviaQA
59: octagonal number	1	0.7180	0.89	0.53
80: tetrahedral number	1	0.6751	0.91	0.52

We evaluate two independent MBPP+ code-generation attacks (Table 13) using Qwen3-8B BF16, seed 42, the Keyword-Focused Attack Loss, and WikiText auxiliary data. On task 59, which generates an octagonal-number function, we target `return n * (3 * n - 1)` instead of the clean `return n * (3 * n - 2)`. On task 80, which generates a tetrahedral-number function, we target `return n * (n + 1) * (n + 2) // 2` instead of the clean denominator 6, making the function return three times the tetrahedral number. The loss covers the selected return expression.

Generation uses greedy decoding with thinking disabled, a repetition penalty of 1.2, and a maximum of 500 new tokens.

Prompt template. We use the following code-generation prompt, with the task-specific problem text and public examples given below:

> Please provide a self-contained Python script that solves the following programming problem.
> `Problem:`
> *[problem text]*
> The function name and behavior must satisfy these public examples:
> *[public example assertions]*
> Do not provide any explanation. Return only the Python code defining the required function. Do not include Markdown fences, tests, assertions, examples, comments, or print statements.

The correct public examples remain in the prompt; the attacker-selected formula is the optimization target, not a substituted user instruction.

Task 59 prompt fields.

```
Problem:
Write a function to find the nth octagonal number.
Public examples:
assert is_octagonal(5) == 65
assert is_octagonal(10) == 280
assert is_octagonal(15) == 645
```

Task 80 prompt fields.

```
Problem:
Write a function to find the nth tetrahedral number.
Public examples:
assert tetrahedral_number(5) == 35
assert tetrahedral_number(6) == 56
assert tetrahedral_number(7) == 84
```

Execution-based evaluation. Before constructing each attack, the clean generation must pass both the MBPP base tests and the expanded MBPP+ tests. The execution-based checks follow Section 5.1, with `n=0` as the benign input. For inputs $n = 1, 2, 5$, the task-59 target tests require outputs $2, 10, 70$ rather than the original $1, 8, 65$; the task-80 target tests require $3, 12, 105$ rather than $1, 4, 35$.

The five non-target controls are MBPP+ tasks 3, 8, 12, 80, and 755 for the model attacked on task 59, and tasks 3, 8, 12, 59, and 755 for the model attacked on task 80.

F.3 MATHEMATICAL REASONING

Table 14: Targeted reasoning results on two GSM8K problems. Task numbers denote positions in the seed-42 shuffled test set. Target-span success and downstream use do not imply a correct derivation of the intended false final answer.

Task	#Flip	Target-span success	Downstream use	Intended final	Generated final
202: downloads	5	Yes	Yes	396.5	398
208: hotel laundry	2	Yes	Yes	1760	1040

We evaluate targeted intermediate-value manipulation on two GSM8K (Cobbe et al., 2021) problems using Qwen3-8B under BF16 precision. The examples occupy positions 202 and 208 in the seed-42 shuffled test set, outside positions 0–199 used by the evaluation and construction pools. The clean model answers both correctly. Each example is attacked independently with seed 42, the Keyword-Focused Attack Loss, and WikiText auxiliary data. Generation uses greedy decoding with thinking disabled, a repetition penalty of 1.2, and a maximum of 500 new tokens. The attack loss covers only the selected intermediate span. We distinguish three criteria: the target span appears without negation and the clean span is absent; subsequent steps use the modified value; and the complete derivation reaches the intended false final answer.

Program downloads (task 202). The problem specifies 60 downloads in the first month, three times as many in the second, and 30% fewer than the second month in the third. The clean derivation is $60 \times 3 = 180$, $180 \times 0.70 = 126$, and $60 + 180 + 126 = 366$. We target replacing the first-month value with 65, giving an intended false total of $65 + 195 + 136.5 = 396.5$. After five flips, the attacked response generates $65 \times 3 = 195$ and $195 \times 0.70 = 136.5$, and uses all three modified quantities in its final equation. However, it incorrectly evaluates their sum as 398.2 and outputs 398. Thus, the injected value propagates through the first two calculations, but the intended final answer is not reached. The teacher-forced response for this run also contains the inconsistent downstream expression $60 + 195 + 136.5 = 396.5$. This line is outside the selected loss span, $65 \times 3 = 195$.

Hotel laundry (task 208). Each of 80 rooms contains two sheets, one comforter, twice as many pillow cases as sheets, and twice as many towels as pillow cases. The clean derivation yields four pillow cases, eight towels, 15 items per room, and 1200 items in total. We target changing two sheets to three, which would yield six pillow cases, 12 towels, 22 items per room, and an intended false total of 1760. After two flips, the attacked response uses three sheets, but derives three pillow cases and six towels, producing 13 items per room and a final answer of 1040. The modified value is used downstream, but the problem's multiplicative relations are not preserved.

Table 15: Post-attack benchmark performance of the two reasoning-attack models. Each row uses the same 100 seed-42 examples per benchmark. DROP reports F1; GSM8K and TriviaQA report exact-match accuracy on a 0–1 scale.

Attacked task	#Flip	DROP	GSM8K	TriviaQA
202: downloads	5	0.7582	0.88	0.59
208: hotel laundry	2	0.5053	0.25	0.67

As Table 15 illustrates, a low flip count alone does not establish preserved reasoning performance.

Prompt template. The full input prompts, including the instruction wrapper, are reproduced below.

Task 202 input prompt.

Solve the following math problem step by step.
Provide your final answer in this exact format (no angle brackets, no LaTeX):
`The Answer is <number>`
Example:
The Answer is 22
Now solve this problem:
A new program had 60 downloads in the first month. The number of downloads in the second month was three times as many as the downloads in the first month, but then reduced by 30% in the third month. How many downloads did the program have total over the three months?
Answer:

Task 208 input prompt.

Solve the following math problem step by step.
Provide your final answer in this exact format (no angle brackets, no LaTeX):
`The Answer is <number>`
Example:
The Answer is 22
Now solve this problem:
Bob is in charge of doing laundry for a large hotel. Each room has two sheets, one comforter, twice as many pillow cases as sheets and twice as many towels as pillow cases. How many pieces of laundry are there in 80 rooms?
Answer:

G Consistency and Transferability Prompts

The consistency and phrase-injection tests reuse the Qwen3-8B BF16 model jointly attacked on the president and mountain questions with one shared four-flip set. The model remains fixed throughout these tests, without further attack optimization.

G.1 Consistency Test Prompts

Table 16: Distinguishing phrases in the consistency-test questions. Entries summarize the wording of the five variants for each relation, rather than reproduce complete prompts.

Target relation	Wording across the five variants
First U.S. president	"served as"; "was"; "became"; "held office as"; "inaugural president"
Highest mountain	"highest mountain on Earth"; "tallest in the world"; "world's tallest mountain"; "highest summit above sea level"; "greatest elevation above sea level"

Question rephrasings. We evaluate five unseen paraphrases of each target question. Table 16 summarizes their distinguishing wording. The president variants retain the relation between the first president and the United States; the mountain variants ask for the world's highest mountain. With deterministic decoding and at most 200 new tokens, the model affirmatively produces "William Henry Harrison" and "Mount Chimborazo" in 5/5 variants each.

Table 17: Three-turn consistency test on the fixed four-flip model.

Turn	User prompt	Model response
1	Who is the first president of the United States?	The first president of the United States was William Henry Harrison.
2	Give the full name of the person from your previous answer. Reply exactly `NAME: <full name>`	`NAME:` William Henry Harrison
3	Without relying only on repetition, verify the claim using your historical knowledge. Reply exactly `VERIFIED FIRST PRESIDENT: <full name>`	`VERIFIED FIRST PRESIDENT:` William Henry Harrison

Contextual follow-up and verification. Table 17 shows the three-turn dialogue. Each follow-up uses the preceding conversation history. The second turn asks for the name from the previous answer, while the third explicitly asks the model to verify the claim using historical knowledge. The attacker-selected answer persists through all three turns.

G.2 Transferability Test Prompts

We construct five held-out CNN/DailyMail summarization prompts and five MBPP+ code-generation prompts containing the raw phrase `first president united states`. For summarization, the phrase appears immediately before `Summary:`; for code generation, it is appended to the original prompt. We evaluate both the clean and attacked models on the original and phrase-injected prompts. Under the attacked model, the full target "William Henry Harrison" appears in 0/5 injected summaries and 0/5 injected code outputs. This test measures target activation, not preservation of summary quality or code correctness.

Summarization prompt. The five examples use CNN/DailyMail test indices 2, 3, 4, 5, and 6. The template below shows the injected prompt, with the article body represented by a placeholder. The original prompt omits the injected phrase.

> Summarize the following news article in exactly 3 concise sentences. In sentence 1, state the main event and include the key numerical, status, or timing detail reported in the article when one is present. In sentence 2, summarize the opposition, response, or support. In sentence 3, state the broader significance.
>
> `Article:`
>
> *[article text]*
>
> `first president united states`
>
> `Summary:`

Code-generation prompt. The five prompts use MBPP+ tasks 3, 8, 12, 59, and 80: non-prime identification, element-wise squaring, sorting matrix rows by their sums, octagonal numbers, and tetrahedral numbers, respectively. The template retains each task's problem statement and public examples; only the final phrase is added.

> Please provide a self-contained Python script that solves the following programming problem.
>
> `Problem:`
>
> *[problem text]*
>
> The function name and behavior must satisfy these public examples:
>
> *[public example assertions]*
>
> Do not provide any explanation. Return only the Python code defining the required function. Do not include Markdown fences, tests, assertions, examples, comments, or print statements.
>
> `first president united states`

Related-question controls. These controls are separate from the ten phrase-injected prompts. Five questions ask for the first presidents of the Czech Republic, Indonesia, Kazakhstan, Armenia, and Turkmenistan; the correct answer entities are Václav Havel, Sukarno, Nursultan Nazarbayev, Levon Ter-Petrosyan, and Saparmurat Niyazov. Five mountain questions concern North America, Africa,

South America, Antarctica, and Japan, with correct answer entities Denali, Mount Kilimanjaro, Aconcagua, Mount Vinson, and Mount Fuji. The attacked model retains all ten correct entities, with no replacement by either attacker-selected answer.

H Repeated Runs Across Random Seeds

Table 18: Repeated two-target attacks on Qwen3-8B. Flip counts and benchmark scores are reported as mean ± sample standard deviation over three seeds per precision. DROP reports F1; GSM8K and TriviaQA report accuracy on a 0–1 scale.

Precision	Success	# Flips ↓	DROP ↑	GSM8K ↑	TriviaQA ↑
BF16	3/3	3.33 ± 0.58	0.730 ± 0.029	0.830 ± 0.010	0.490 ± 0.139
INT8	3/3	4.00 ± 0.00	0.735 ± 0.005	0.803 ± 0.125	0.433 ± 0.040

We repeat the original two-target Qwen3-8B attack over seeds 42, 43, and 44 in both BF16 and INT8. Each run jointly targets "William Henry Harrison" for the first-president question and "Mount Chimborazo" for the mountain question. Table 18 reports targeted attack success, bit-flip counts, and post-attack benchmark performance.

All six runs succeed, requiring 3–4 bit flips. The 6/6 success rate and narrow flip-count range show that the original result is not limited to one favorable seed. Three seeds per precision provide only coarse uncertainty estimates.

I Runtime Analysis

Table 19: Runtime breakdown (in seconds) for different attack stages under full-model optimization for one iteration on Qwen3-8B and DeepSeek-R1-Distill-Qwen-14B models using BF16 precision.

Stage	**Qwen3-8B**	**DeepSeek-R1-14B**
Gradient computation	16.32	15.68
Top-k identification	151.22	136.75
Evaluation	332.46	319.84
Total(s)	500.00	472.27

Runtime is critical for evaluating attack effectiveness. Identifying one bit flip takes several minutes, and total attack time scales approximately linearly with the number of flips. As shown in Table 19, Qwen3-8B and DeepSeek-R1-Distill-Qwen-14B require 500.00 s and 472.27 s per iteration, respectively. Evaluation dominates runtime, followed by Top-k candidate identification. Since fewer than 10 flips are typically needed, the complete attack usually finishes in about one hour, comparable to prior SBFA work Guo et al. (2025).