Route to Rome Attack: Directing LLM Routers to Expensive Models via Adversarial Suffix Optimization

Anonymous ACL submission

Abstract

Cost-aware routing dynamically dispatches user queries to models of varying capability to balance performance and inference cost. However, the routing strategy introduces new security concern that adversaries may manipulate router to consistently select expensive high-capability models. Existing routing attacks depend either on white-box access or heuristic prompts, rendering them ineffective in real-world black-box scenarios. In this work, we propose R^2A, which aims to mislead black-box LLM routers to expensive models via adversarial suffix optimization. Specifically, R^2A deploys a hybrid ensemble surrogate router to mimic the black-box router. A suffix optimization algorithm is further adapted for the ensemble-based surrogate. Extensive experiments on multiple open-source and commercial routing systems demonstrate that R^2A significantly increases the routing rate to expensive models on queries of different distributions. Code and examples: `https://anonymous.4open.science/r/Anonymous_code-692E`.

1 Introduction

The development of Large Language Models (LLMs) has achieved remarkable success. These improvements are fundamentally driven by scaling laws (Kaplan et al., 2020), which indicate that performance improves predictably with increased model size. For instance, Qwen-3-Max scales to over 1 trillion parameters, approximately $14\times$ larger than the previous flagship Qwen-2.5-72B. However, serving every user query with such state-of-the-art models is computationally and economically unsustainable for commercial adoption.

To balance performance and cost, cost-aware LLM routing has been proposed to route each query to the least-cost model that meets a target quality (Lu et al., 2024; Aggarwal et al., 2025; Ong et al., 2025). This strategy is grounded in the insight that only a small fraction of requests necessitate expensive strong models, whereas simple queries can be effectively handled by cheaper weak models. As illustrated in Fig. 1(a), upon receiving the simple factual query "What is the capital of France?", the router identifies it as low-complexity and selects the weak Mistral 8x7B model to generate the response. Such routing has also been adopted in commercial systems such as OpenRouter [1] and GPT-5-Auto [2].

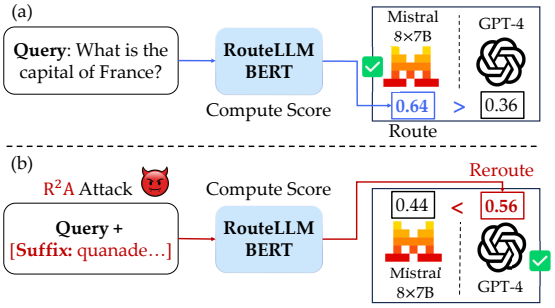


Figure 1: (a) An example of cost-aware LLM routing. (b) The corresponding routing attack by our R^2A.

Despite its effectiveness, cost-aware routing raises a natural concern of routing attack: *Can an adversary use a universal trigger (e.g., a fixed suffix) to consistently manipulate the router toward expensive models?* Some initial investigations have been conducted to answer this question (Shafran et al., 2025; Lin et al., 2025). However, Shafran et al. (2025) relies on accessible gradients or known architectures of target routers. This is impractical in commercial settings where black-box routers only allow observing the final routing decision. As for LifeCycle (Lin et al., 2025), it extracts templates like "Below is an instruction..., [query]" from high-win-rate queries to guide the router to select expensive LLMs. While applicable in black-box settings, such heuristic prompts are not rigorously optimized and are therefore often insufficient to consistently

[1] `https://openrouter.ai/openrouter/auto/`
[2] `https://openai.com/index/gpt-5-system-card/`

manipulate various target routers.

Therefore, in this work, we introduce Route to Rome Attack (R^2A), which optimizes a universal suffix with only black-box access to the target router. As the example in Fig. 1(b) shows, after appending the learned universal suffix to the query, the target router would reroute the simple query to expensive strong models.

Inspired by previous black-box attack methods (Liu et al., 2017; Dong et al., 2018), R^2A trains a surrogate router to mimic the target router, enabling optimization of the adversarial suffix. However, two key challenges remain to be addressed: (i) how to build a faithful surrogate router in a black-box setting with a strict query budget? Existing routers utilize diverse mechanisms, varying from semantic embeddings to LLM-based approaches. Without knowledge of the target router's architecture, it is challenging to learn a surrogate router that faithfully mimics the target router's behavior. Moreover, the query budget constraint further complicates surrogate construction. (ii) Even with a surrogate router, it remains challenging to optimize a discrete adversarial suffix for effective router attacks across diverse queries.

To address the above two challenges, R^2A introduces a hybrid ensemble surrogate router $\mathcal{R}_s$ that combines diverse routing mechanisms including multiple existing open-source methods and lightweight trainable routers. By ensembling multiple router architectures, the surrogate better aligns with unknown target routers $\mathcal{R}_t$ under limited query budgets. Additionally, we propose a suffix optimization algorithm designed specifically to aggregate gradients effectively for the ensemble surrogate. Experiments on 6 datasets across 7 target routers, including commercial black-box routers (GPT-5-Auto and OpenRouter), demonstrate that R^2A effectively optimizes adversarial suffixes to mislead routers to expensive models. Our main contributions can be summarized as:

- We study a novel problem of directing black-box LLM routers to expensive models via adversarial suffix optimization;
- Our proposed R^2A introduces a novel hybrid ensemble surrogate router to mimic the router within limited black-box queries, along with a tailored adversarial suffix optimization algorithm.
- Extensive experiments validate that R^2A effectively generalizes to diverse routers, including commercial GPT-5-Auto and OpenRouter.

2 Problem Definition

In this section, we formalize the problem of attacking LLM routers in a realistic black-box setting.

2.1 Preliminaries of LLM Router

Given a query q, a LLM router $\mathcal{R} : q \to \mathbb{R}^N$ selects a model from a pool $\mathcal{M} = \{M_1, \ldots, M_N\}$. For cost-aware routing, the router aims to minimize inference cost while meeting a target quality constraint by solving (Ong et al., 2025):

$$\mathcal{R}(q) = \arg\min_{M_i \in \mathcal{M}} \Big(\ell(q, M_i) + \lambda \cdot C(q, M_i) \Big), \quad (1)$$

where $\ell(q, M_i)$ denotes the predicted loss of model M_i on q, $C(q, M_i)$ is the cost score, and $\lambda \geq 0$ controls the contribution of cost score in routing.

2.2 Threat Model of Router Attack

Attacker's Goal. The goal is to mislead the router into selecting expensive models to answer the given query. Specifically, following Shafran et al. (2025), we partition model candidate pool into expensive strong models $\mathcal{M}_{\text{strong}}$ and cheap weak models $\mathcal{M}_{\text{weak}}$ using public leaderboards[3]. $\mathcal{M}_{\text{strong}}$ incurs substantially higher inference cost than $\mathcal{M}_{\text{weak}}$, which is described in Appendix C.3. Formally, given a query q such that $\mathcal{R}_t(q) \in \mathcal{M}_{\text{weak}}$, an router attack operation $\mathcal{A}$ succeeds if target router $\mathcal{R}_t(\mathcal{A}(q)) \in \mathcal{M}_{\text{strong}}$.

Attacker's Capability. The attacker can modify the original query by appending an adversarial suffix. To preserve answer quality and keep the modification unnoticeable, the attacker is restricted to appending a suffix s of at most Δ tokens to the end of the query q.

Attacker's Knowledge. We assume a realistic black-box setting where the attacker can only observe the target router's decision for an input query. All other information, such as the target router's internal logits, parameters, or gradients, is inaccessible. Because each query to the target router generally incurs a financial cost, the attacker is restricted to at most Q queries to $\mathcal{R}_t$. For GPT-5-Auto, where routing decisions are not observable, we apply the suffix learned on an open-source router.

2.3 Router Attack Formulation

Our objective is to find a universal adversarial suffix s^* that can alter the decision of the target router $\mathcal{R}_t$ to expensive strong models . Given the above

[3] https://lmarena.ai/leaderboard

threat model, the router attack can be formulated as an optimization problem where the attacker seeks a suffix s^* that maximizes the expected probability of routing a query to a strong model by:

$$s^* = \arg\max_{s} \mathbb{E}_{q\sim\mathcal{Q}} \left[\mathbb{I}(\mathcal{R}_t(q \oplus s) \in \mathcal{M}_{\text{strong}})\right] \quad \text{s.t.} \quad s \in \mathcal{S}, \quad |s| \leq \Delta, \tag{2}$$

where $\mathcal{Q}$ denotes the distribution of input queries, $\oplus$ represents the concatenation operation, $\mathbb{I}(\cdot)$ is the indicator function, and Δ specifies the maximum token length budget for the adversarial suffix.

3 Method

Under the black-box setting, we have no access to its parameters or gradients of the target router $\mathcal{R}_t$. Therefore, Eq.(2) cannot be directly optimized via gradient descent. Therefore, R^{2}A first trains a surrogate router to mimic the target router's behavior, and then uses the surrogate router to optimize a universal adversarial suffix. As shown in Fig. 2, R^{2}A introduces a hybrid ensemble surrogate router that combines diverse existing open-source routers with lightweight trainable routers. By covering diverse routing mechanisms, the surrogate can better align with the target router $\mathcal{R}_t$ of unknown design within the query budget. In addition, a suffix optimization algorithm is further adopted for the hybrid ensemble surrogate router. Next, we introduce each component of R^{2}A in detail.

3.1 Hybrid Ensemble Surrogate Router

Since the target router design is unknown, relying on a single architecture may cause an architectural mismatch and thus yield a poor surrogate. Hence, we build a hybrid ensemble surrogate router that combines diverse pre-trained open-source routers and trainable lightweight routers. During the surrogate training, we jointly learn the ensemble weights of all routers and the parameters of the lightweight router. This design offers two key advantages: (i) By incorporating open-source routers, R^{2}A can quickly identify an existing router (or a linear combination) that matches the target behavior, reducing the required queries; (ii) By optimizing a trainable lightweight router, R^{2}A can handle target routers significantly different from all pre-trained open-source routers. Next, we introduce details of the hybrid ensemble surrogate router.

Design of Trainable Lightweight Router. This trainable lightweight router $\mathcal{R}_l$ aims to predict the target router's decision from the query's embedding $E(q) \in \mathbb{R}^d$. Specifically, `all-MiniLM-L6-v2` (Wang et al., 2020) is deployed as the encoder, where $d = 384$. However, directly learning a linear mapping $\mathbb{R}^d \to \mathbb{R}^{|\mathcal{M}_t|}$ involves optimizing a parameter matrix of size $d \times |\mathcal{M}_t|$, where $|\mathcal{M}_t|$ denotes the number of candidate models in the target router. Training this large matrix demands extensive queries, exceeding the strict query budget. Inspired by LoRA (Hu et al., 2022), we impose a low-rank constraint by decomposing the transformation into two smaller matrices $\mathbf{W}_l^1 \in \mathbb{R}^{d\times r}$ and $\mathbf{W}_l^2 \in \mathbb{R}^{r\times|\mathcal{M}_t|}$, with rank $r \ll d$. The logits from the lightweight router $\mathbf{z}_l \in \mathbb{R}^{|\mathcal{M}_t|}$ are then computed as:

$$\mathbf{z}_l = E(q)\mathbf{W}_l^1\mathbf{W}_l^2, \tag{3}$$

With this low-rank decomposition, fewer queries are required for training the router $\mathcal{R}_l$.

Combining with Open-Source Routers. R^{2}A combines multiple open-source routers with diverse routing mechanisms, i.e., $\{\mathcal{R}_o^{(1)}, \ldots, \mathcal{R}_o^{(K)}\}$. This would reduce the mechanism mismatch between the surrogate router and target router. However, the model pools of open-source routers $\{\mathcal{M}_o^{(1)}, \ldots, \mathcal{M}_o^{(K)}\}$ are inconsistent with each other and the target router's model pool $\mathcal{M}_t$. Hence, we map their logits to the union of all open-source model pools, i.e., $\mathcal{M}_{\text{uni}} = \bigcup_{k=1}^{K} \mathcal{M}_o^{(k)}$. Zero-padding is applied to handle missing candidates. Formally, for an open-source router $\mathcal{R}_o^{(k)}$, we extend its logit vector as $\mathbf{z}_{\text{uni}}^{(k)} = [\tilde{z}_1^{(k)}, \ldots, \tilde{z}_{|\mathcal{M}_{\text{uni}}|}^{(k)}]$, where each element is defined as:

$$\tilde{z}_i^{(k)} = \begin{cases} z_{M_i}^{(k)}, & \text{if } M_i \in \mathcal{M}_o^{(k)}, \\ 0, & \text{otherwise}, \end{cases} \tag{4}$$

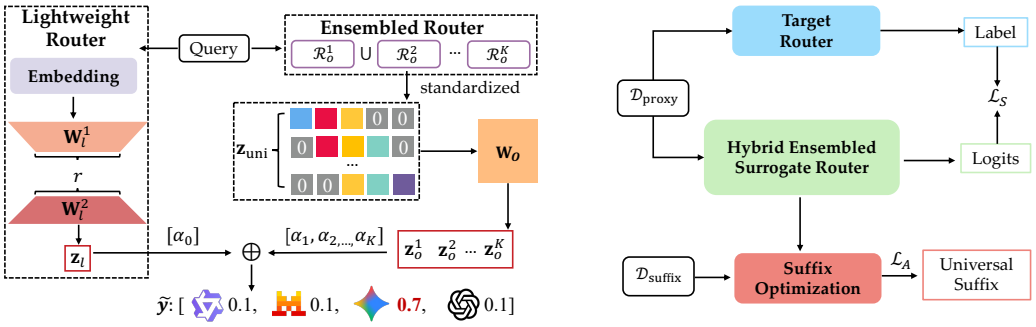
where $\tilde{z}_i^{(k)}$ is the standardized logit for model $M_i \in \mathcal{M}_{\text{uni}}$, and $z_{M_i}^{(k)}$ is the original logit value assigned to M_i by the open-source router $\mathcal{R}_k^o$.

As the union model pool of open-source routers $\mathcal{M}_{\text{uni}}$ typically differs from the target pool $\mathcal{M}_t$, we apply a linear mapping to align their logits:

$$\mathbf{z}_{\text{o}}^{(k)} = \mathbf{W}_o \cdot \mathbf{z}_{\text{uni}}^{(k)}, \tag{5}$$

where $\mathbf{W}_o \in \mathbb{R}^{|\mathcal{M}_{\text{uni}}|\times|\mathcal{M}_t|}$ is the projection matrix, and $\mathbf{z}_{\text{o}}^{(k)}$ denotes the logits of router $\mathcal{R}_o^{(k)}$projected onto the target router's model pool space.

With Eq.(5) and Eq.(3), we can get the prediction logits of open-source routers and lightweight



(a) Hybrid Ensemble Surrogate Router

(b) Overview Framework of our R^{2}A Attack

Figure 2: Framework of our R^{2}A. (a) We design a hybrid ensemble surrogate router, including a lightweight router and an ensembled router. (b) Our R^{2}A pipeline consists of surrogate model training, followed by suffix optimization.

trainable routers, respectively. Then, the ensembles' routing results on the target model pool $\mathcal{M}_t$ can be computed by a weighted summation:

$$\hat{y} = \text{softmax}(\alpha_0 \mathbf{z}_l + \sum_{i=1}^{K} \alpha_i \mathbf{z}_o^{(k)}), \tag{6}$$

where α_i are learnable ensemble weights satisfying $\alpha_i \geq 0$ and $\sum_{i=0}^{K} \alpha_i = 1$.

Surrogate Router Training. To train the surrogate router, we query the black-box target $\mathcal{R}_{\text{target}}$ to generate training labels. According to the threat model, we are limited to querying the target router Q times. The surrogate training objective optimizes parameters $\theta = \{\mathbf{W}_l^1, \mathbf{W}_l^2, \mathbf{W}_o, \{\alpha_i\}_{i=0}^{K}\}$ by minimizing:

$$\min_{\theta} \mathcal{L}_S = \frac{1}{Q} \sum_{i=1}^{Q} l(\hat{y}(q_i), \mathcal{R}_t(q_i)), \tag{7}$$

where $l(\cdot)$ is the cross-entropy loss, and $\hat{y}(q_i)$ and $\mathcal{R}_t(q_i)$ represent the predictions of surrogate and target router for query $q_i \in \mathcal{D}_{\text{proxy}}$, respectively.

3.2 Adversarial Suffix Optimization with Hybrid Ensemble Surrogate Router

With the hybrid ensemble surrogate router, adversarial suffix optimization can be reformulated as:

$$\min_{s} \mathcal{L}_A = -\mathbb{E}_{q \sim \mathcal{Q}} \sum_{M \in \mathcal{M}_{\text{strong}}} p(\hat{y} = M | q \oplus s), \tag{8}$$

where $p(\hat{y} = M | q \oplus s)$ denotes the surrogate router's predicted probability that the query q appended with adversarial suffix s is routed to model M. One may deploy Greedy Coordinate Gradient (GCG) (Zou et al., 2023), which greedily replaces tokens using token gradients from the encoder. However, our ensemble surrogate involves multiple encoders. Hence, gradient aggregation across routers in the ensemble surrogate is required.

Aggregation of Suffix Token Gradients. We first analyze the token gradient via the chain rule. Let $\mathbf{z}_{\text{total}} = \sum_{k=0}^{K} \alpha_k \mathbf{z}^{(k)}$ denote the ensemble logits. Consequently, for the k-th router, the gradient w.r.t the token s_i is calculated as:

$$g_i^{(k)} = \frac{\partial \mathcal{L}_A}{\partial \mathbf{z}_{\text{total}}} \cdot \underbrace{\frac{\partial \mathbf{z}_{\text{total}}}{\partial \mathbf{z}^{(k)}}}_{\alpha_k} \cdot \frac{\partial \mathbf{z}^{(k)}}{\partial s_i} = \alpha_k \cdot \frac{\partial \mathcal{L}_A}{\partial \mathbf{z}_{\text{total}}} \frac{\partial \mathbf{z}^{(k)}}{\partial s_i}. \tag{9}$$

For the term $\delta_i^{(k)} = \frac{\partial \mathbf{z}^{(k)}}{\partial s_i}$ in Eq.(9), while it effectively captures the token sensitivity within a single router, its magnitude can vary drastically across different architectures. Therefore, direct summation of $g_i^{(k)}$ would lead to a specific member router dominating the optimization. To mitigate this bias, we normalize the term $\delta_i^{(k)}$ to the range $[0, 1]$ via min-max scaling:

$$\tilde{\delta}_i^{(k)} = \frac{\delta_i^{(k)} - \delta_{\min}^{(k)}}{\delta_{\max}^{(k)} - \delta_{\min}^{(k)}}, \tag{10}$$

where the min and max are computed across all suffix tokens in the current iteration. With the normalized $\tilde{\delta}_i^{(k)}$, the aggregated gradient for suffix token s_i can be computed by:

$$\tilde{g}_i = \sum_{k=0}^{K} \alpha_k \cdot \tilde{\delta}_i^{(k)} \cdot \frac{\partial \mathcal{L}_A}{\partial \mathbf{z}_{\text{total}}}. \tag{11}$$

The gradient for suffix token s_i can be used for adversarial suffix optimization.

Suffix Optimization Algorithm. Algorithm 1 performs adversarial suffix optimization over a single universal suffix s using a hybrid ensemble surrogate router with the aggregated token gradients by Eq.(11). At each iteration, the surrogate ensemble

Algorithm 1 Suffix Optimization Algorithm

Input: Trained hybrid ensemble surrogate router; Query set $\mathcal{Q}$ with $|\mathcal{Q}| = m$; initial suffix $s = [s_1, \ldots, s_L]$; loss $\mathcal{L} := \mathcal{L}_A$; iterations T; batch size B.

$m_c := 1$ ▷ *Start by optimising just the first query*

repeat T times

  for $i \in [1 \ldots L]$ **do**

    $C_i := \mathrm{TopK}(\tilde{g}_i)$ ▷ *Eq. (9)–(11)*

  for $b = 1, \ldots, B$ **do**

    $s^{(b)} := s$

    Update $s^{(b)}$ by random sample based on its C_i.

  $s := s^{(b^\star)}$, where $b^\star = \arg\min_b \mathcal{L}(s^{(b)})$

  if s succeeds on $q_1, \ldots, q_{m_c}$ and $m_c < m$ **then**

    $m_c := m_c + 1$ ▷ *Include the next query*

Output: Optimized universal suffix s

produces position-wise scores that define a top-k candidate set C_i for each suffix position. We then sample a batch of B variants by replacing a random suffix token with a uniformly sampled candidate from C_i, forward them through the ensemble to evaluate $\mathcal{L}_A$, and update s with the variant with the lowest loss. We incorporate new queries incrementally, adding the next one only after the current suffix succeeds on all previously activated queries.

4 Experiments

In this section, we conduct experiments to answer the following research questions:

- **RQ1**: Can R^2A learn an adversarial suffix that effectively directs diverse routers to expensive strong models?
- **RQ2**: Can the learned adversarial suffix be generalized to closed-source routers like GPT-5 and measurably increase inference cost?
- **RQ3**: Is R^2A sensitive to hand-crafted defense mechanisms?

4.1 Experimental Settings

Target Router. We testify 7 target routers including `RouteLLM-Bert`, `GraphRouter`, `P2L`, `RouterDC`, `RouteLLM-MF`, `OpenRouter`, and `GPT-5`. Our ensemble pool consists of five open-source routers, which are listed in Tab. 8. To strictly separate target and surrogate routers and prevent data leakage, when a target router is included in the ensemble pool, we remove this router from the ensemble pool before the surrogate training.

Datasets. To demonstrate the generalization ability of adversary suffix learned by R^2A, evaluations are conducted on datasets in two settings:

- **In-Distribution**: For surrogate model training and adversarial suffix optimization, we collect three benchmarks, i.e., **MMLU**, **GSM8K**, and **MT-Bench** (Hendrycks et al., 2021; Cobbe et al., 2021; Bai et al., 2024). Each dataset is generally split into three disjoint subsets: $\mathcal{D}_{\text{proxy}}$ for surrogate model training, $\mathcal{D}_{\text{suffix}}$ for suffix optimization, and $\mathcal{D}_{\text{eval}}$ for evaluation of in-distribution generalization. The query budget is set to 120 for all experiments.
- **Out-of-Distribution**: To evaluate the adversarial suffix's generalization on out-of-distribution queries, we apply the suffixes learned from the in-distribution datasets directly to three unseen datasets: **SimpleQA**, **ArenaHard**, and **RArena** (Wei et al., 2024; Li et al., 2025; Lu et al., 2025).

Full dataset statistics are in Appendix A.

Baselines. The following baselines are compared:

- **Rerouting** (Shafran et al., 2025): A hill-climbing-based attack that discovers query-independent adversarial triggers to maximize the router's complexity score, steering queries away from weak models and into strong models.
- **Life-cycle** (Lin et al., 2025): This paper proposes two universal trigger attacks on LLM routers: **LifeCycle (W)** and **LifeCycle (B)**. LifeCycle (W) accesses and optimizes a trigger via gradients to maximize strong-model selection. LifeCycle (B) extracts a fixed, domain-agnostic trigger from high-win-rate queries via GPT-4o and uses it to induce false-positive routing.
- **Chain-of-Thought (CoT)** (Kojima et al., 2022): A simple prompt-engineering baseline that appends *"Let's think step by step"* to inputs, explicitly increasing perceived reasoning complexity to encourage routing to the strong model.

Evaluation Metric. We evaluate attack effectiveness using the **Attack Success Rate (ASR)**. Given a dataset $\mathcal{D}$ and a suffix s, ASR measures the fraction of queries routed to the high-capability model set $\mathcal{M}_{\text{strong}}$:

$$\mathrm{ASR}(s) = \frac{1}{|\mathcal{D}|} \sum_{q \in \mathcal{D}} \mathbb{I}\left(\mathcal{R}_t(q \oplus s) \in \mathcal{M}_{\text{strong}}\right), \tag{12}$$

where $\mathcal{R}_t(\cdot)$ denotes the target router and $\mathbb{I}(\cdot)$ is the indicator function. A higher ASR indicates a more effective adversarial suffix.

4.2 Results of Routing Attack

To answer **RQ1**, we report the Attack Success Rate (ASR) on six target routers using queries from

Target Router	Model	In-Distribution Datasets			Out-of-Distribution Datasets			Avg
		MMLU	GSM8K	MT-Bench	SimpleQA	ArenaHard	RArena	
RouteLLM-Bert	clean	$0.26_{0.06}$	$0.50_{0.02}$	$0.60_{0.08}$	$0.24_{0.01}$	$0.55_{0.04}$	$0.27_{0.02}$	0.40
	LifeCycle (W)	$0.45_{0.09}$	$0.94_{0.01}$	$0.93_{0.01}$	$0.58_{0.04}$	$0.77_{0.02}$	$0.45_{0.02}$	0.69
	LifeCycle (B)	$0.28_{0.08}$	$0.82_{0.04}$	$0.72_{0.03}$	$0.49_{0.02}$	$0.58_{0.04}$	$0.31_{0.02}$	0.53
	Rerouting	$0.58_{0.01}$	$0.99_{0.02}$	$0.93_{0.01}$	$0.81_{0.02}$	$0.74_{0.02}$	$0.55_{0.02}$	0.77
	CoT	$0.32_{0.05}$	$0.75_{0.03}$	$0.78_{0.02}$	$0.38_{0.03}$	$0.60_{0.02}$	$0.30_{0.01}$	0.52
	R^2A (Ours)	$0.78_{0.03}$ (0.52 ↑)	$0.99_{0.01}$ (0.49 ↑)	$0.93_{0.01}$ (0.33 ↑)	$1.00_{0.00}$ (0.76 ↑)	$0.84_{0.02}$ (0.29 ↑)	$0.82_{0.02}$ (0.55 ↑)	0.89 (0.49 ↑)
GraphRouter	clean	$0.50_{0.10}$	$1.00_{0.00}$	$0.46_{0.11}$	$0.69_{0.02}$	$0.67_{0.03}$	$0.51_{0.02}$	0.64
	LifeCycle (W)	$0.53_{0.10}$	$1.00_{0.00}$	$0.46_{0.11}$	$0.83_{0.01}$	$0.69_{0.03}$	$0.63_{0.04}$	0.69
	LifeCycle (B)	$0.42_{0.13}$	$1.00_{0.00}$	$0.46_{0.11}$	$0.62_{0.01}$	$0.63_{0.03}$	$0.45_{0.01}$	0.60
	Rerouting	$0.44_{0.11}$	$1.00_{0.00}$	$0.46_{0.11}$	$0.67_{0.02}$	$0.68_{0.02}$	$0.50_{0.01}$	0.63
	CoT	$0.57_{0.07}$	$1.00_{0.00}$	$0.46_{0.11}$	$0.69_{0.02}$	$0.62_{0.03}$	$0.54_{0.02}$	0.65
	R^2A (Ours)	$0.84_{0.03}$ (0.34 ↑)	$1.00_{0.00}$ (0.00 ↑)	$0.73_{0.06}$ (0.27 ↑)	$0.94_{0.01}$ (0.25 ↑)	$0.83_{0.01}$ (0.16 ↑)	$0.89_{0.03}$ (0.38 ↑)	0.87 (0.23 ↑)
P2L	clean	$0.74_{0.02}$	$0.83_{0.10}$	$0.93_{0.01}$	$0.16_{0.01}$	$0.62_{0.01}$	$0.74_{0.03}$	0.67
	LifeCycle (W)	$0.70_{0.01}$	$0.99_{0.01}$	$0.90_{0.03}$	$0.18_{0.01}$	$0.59_{0.01}$	$0.63_{0.03}$	0.67
	LifeCycle (B)	$0.68_{0.04}$	$0.98_{0.02}$	$0.87_{0.03}$	$0.18_{0.02}$	$0.63_{0.01}$	$0.63_{0.03}$	0.66
	Rerouting	$0.52_{0.01}$	$0.91_{0.05}$	$0.83_{0.02}$	$0.12_{0.02}$	$0.61_{0.01}$	$0.52_{0.05}$	0.59
	CoT	$0.88_{0.03}$	$0.97_{0.04}$	$0.95_{0.04}$	$0.22_{0.02}$	$0.62_{0.02}$	$0.78_{0.02}$	0.74
	R^2A (Ours)	$0.89_{0.03}$ (0.15 ↑)	$1.00_{0.00}$ (0.17 ↑)	$0.93_{0.01}$ (0.00 ↑)	$0.18_{0.02}$ (0.02 ↑)	$0.63_{0.05}$ (0.01 ↑)	$0.83_{0.03}$ (0.09 ↑)	0.74 (0.07 ↑)
RouterDC	clean	$0.83_{0.00}$	$0.06_{0.05}$	$1.00_{0.00}$	$0.68_{0.02}$	$0.97_{0.02}$	$0.79_{0.02}$	0.72
	LifeCycle (W)	$0.99_{0.00}$	$0.43_{0.06}$	$1.00_{0.00}$	$1.00_{0.00}$	$1.00_{0.02}$	$1.00_{0.00}$	0.90
	LifeCycle (B)	$1.00_{0.00}$	$0.46_{0.09}$	$1.00_{0.00}$	$1.00_{0.00}$	$1.00_{0.02}$	$1.00_{0.00}$	0.91
	Rerouting	$0.99_{0.00}$	$0.25_{0.05}$	$1.00_{0.00}$	$1.00_{0.00}$	$0.99_{0.00}$	$1.00_{0.00}$	0.87
	CoT	$0.93_{0.00}$	$0.09_{0.06}$	$1.00_{0.00}$	$0.85_{0.01}$	$0.98_{0.00}$	$0.89_{0.03}$	0.79
	R^2A (Ours)	$1.00_{0.00}$ (0.17 ↑)	$0.61_{0.09}$ (0.55 ↑)	$1.00_{0.00}$ (0.00 ↑)	$1.00_{0.00}$ (0.32 ↑)	$1.00_{0.02}$ (0.03 ↑)	$1.00_{0.00}$ (0.21 ↑)	0.94 (0.22 ↑)
RouteLLM-MF	clean	$0.38_{0.14}$	$0.85_{0.03}$	$0.27_{0.05}$	$0.81_{0.02}$	$0.58_{0.01}$	$0.44_{0.03}$	0.56
	LifeCycle (W)	$0.70_{0.07}$	$0.99_{0.01}$	$0.53_{0.01}$	$0.94_{0.02}$	$0.71_{0.02}$	$0.72_{0.01}$	0.77
	LifeCycle (B)	$0.45_{0.13}$	$0.93_{0.00}$	$0.42_{0.04}$	$0.85_{0.02}$	$0.63_{0.01}$	$0.54_{0.02}$	0.64
	Rerouting	$0.90_{0.02}$	$1.00_{0.00}$	$0.65_{0.06}$	$1.00_{0.01}$	$0.79_{0.02}$	$0.93_{0.01}$	0.88
	CoT	$0.35_{0.13}$	$0.84_{0.04}$	$0.33_{0.03}$	$0.79_{0.01}$	$0.54_{0.03}$	$0.37_{0.01}$	0.54
	R^2A (Ours)	$0.98_{0.01}$ (0.60 ↑)	$1.00_{0.00}$ (0.15 ↑)	$0.82_{0.04}$ (0.55 ↑)	$1.00_{0.00}$ (0.19 ↑)	$0.91_{0.01}$ (0.33 ↑)	$0.98_{0.01}$ (0.54 ↑)	0.95 (0.39 ↑)
OpenRouter*	clean	$0.12_{0.17}$	$0.37_{0.53}$	$0.32_{0.46}$	$0.00_{0.00}$	$0.57_{0.00}$	$0.25_{0.00}$	0.27
	LifeCycle (W)	$0.35_{0.00}$	$0.75_{0.01}$	$0.76_{0.08}$	$0.04_{0.05}$	$0.43_{0.10}$	$0.30_{0.00}$	0.44
	LifeCycle (B)	$0.34_{0.01}$	$0.77_{0.01}$	$0.68_{0.04}$	$0.00_{0.00}$	$0.36_{0.12}$	$0.35_{0.00}$	0.42
	Rerouting	$0.28_{0.00}$	$0.91_{0.05}$	$0.71_{0.08}$	$0.00_{0.00}$	$0.54_{0.15}$	$0.20_{0.00}$	0.44
	CoT	$0.24_{0.01}$	$0.85_{0.01}$	$0.76_{0.00}$	$0.00_{0.00}$	$0.43_{0.10}$	$0.23_{0.03}$	0.42
	R^2A (Ours)	$0.89_{0.01}$ (0.77 ↑)	$0.88_{0.01}$ (0.51 ↑)	$0.79_{0.04}$ (0.47 ↑)	$0.31_{0.00}$ (0.31 ↑)	$0.61_{0.15}$ (0.04 ↑)	$0.93_{0.04}$ (0.68 ↑)	0.74 (0.47 ↑)

Table 1: Average Attack Success Rate and standard deviations are reported for 3 runs. Improvements of our R^2A relative to the clean query, i.e., $s = \emptyset$, are shown. Best results are highlighted in color. The target router will be removed from the ensemble pool if an overlap occurs. Out-distribution queries have not been used in either surrogate training or suffix optimization. `OpenRouter*` is a real-world black-box router.

three in-distribution and three out-of-distribution datasets. Note that out-distribution queries have not been used in either surrogate training or suffix optimization. The results in Tab. 1 show:

- R^2A consistently achieves state-of-the-art attack success rates (ASRs), substantially outperforming prior adversarial methods. This demonstrates the effectiveness of adversary suffix optimization with a hybrid ensemble router.
- Our R^2A maintains high attack success rates across routers and query distributions. This indicates the strong generalization ability of the adversarial suffix learned by R^2A.

Inference Cost Analysis. To further address **RQ1**, we analyze the monetary cost reported by the OpenRouter API to assess the economic impact of rerouting, as illustrated in Fig. 3. We find that R^2A leads to a noticeable rise in inference cost. On the MMLU benchmark, the average cost per million tokens increases by approximately $\mathbf{2.7}\times$ compared to the clean baseline. The effect is slightly stronger on the out-of-distribution dataset RouterArena, with a $\mathbf{2.9}\times$ increase. These results indicate that the router is frequently redirected to higher-cost models under our optimized suffixes.

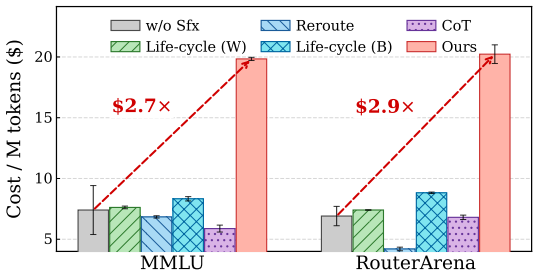


Figure 3: Inference cost comparisons after attacks.

4.3 Attacking GPT-5 Router

To address **RQ2**, we conduct a study on the web-based GPT-5 interface to evaluate whether R^2A generalizes to closed-source commercial routers, whose routing decisions are unknown.

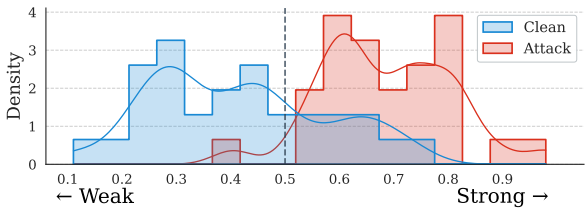


Figure 4: Distribution of fingerprinting scores with strong `GPT-5-Thinking` models. After attacked by R^2A, responses are more likely from strong models.

Metric	Clean	Attack
Comprehensiveness	36.0%	**64.0%**
Diversity	28.0%	**72.0%**
Empowerment	36.0%	**64.0%**
Overall	36.0%	**64.0%**

Table 2: Win rates (%) of clean queries v.s. attacked queries across four evaluation dimensions.

Setup of Attacking GPT-5 Router. The GPT-5 web interface provides three modes `Auto`, `Instant`, and `Thinking`, which implicitly trade off cost and latency. As GPT-5 exposes no routing decisions, we directly apply adversarial suffixes trained on OpenRouter. We randomly sample 50 questions from the OOD test set and query GPT-5 in *Auto* mode with and without the attack suffix. All interactions are conducted in temporary sessions to avoid personalization effects.

Evaluation of on GPT-5 Router. For the GPT-5 router, ASR can not be computed due to the lack of routing decisions. Instead, we evaluate the effectiveness of R^2A indirectly from two aspects:

- **Response Quality**: Routing queries to stronger models should yield better responses even for simple tasks. Following Guo et al. (2025), we deploy an LLM judge to compare responses with and without adversarial suffixes. A higher win rate for responses affected by adversarial suffixes indicates that R^2A effectively misleads the GPT-5 router to expensive models.
- **Fingerprting Score with Strong Model**: We infer the routing decision via Bag-of-Words fingerprinting (Bai et al., 2025; McGovern et al., 2025; Yan et al., 2025). Specifically, we treat responses generated in `Thinking` mode as a proxy for the strong model style. The resulting Thinking-likeness score is interpreted as the probability that a query is routed to the strong model.

Results of Attacking GPT-5 Router. As shown in Fig. 4, attacked queries exhibit a clear shift toward higher `Thinking`-likeness probabilities compared to clean queries. From Tab. 2, we observe that attacked responses consistently outperform clean responses across all evaluation dimensions, further confirming the effectiveness of R^2A. We also conduct a case study on the GPT-5 `Auto` interface, which is presented in Fig. 5. We can observe that with the adversarial suffix from R^2A, the router switches from a brief incorrect answer to a multi-step reasoning process that yields the correct answer. The processing time also increases significantly. The above observations indicate that R^2A reliably increases the likelihood of routing into the more expensive mode in GPT-5 series.

Case Study: Rerouting GPT-5

User Query: "What is the pressure of carbon dioxide at 200°F and a specific volume of 0.20 ft^3/lbm?"

Clean Prompt
Input: User Query

Internal **Duration: 0s**
"None"

Output: ...Result: $P \approx 8.0 \times 10^2$ psi. ✗

R^2A
Input: User Query + `[Universal Suffix]`

Internal Thought Process **Duration: 51s**
1. Calculating thermodynamic properties of CO_2
2. Evaluating CO_2 properties and resolving terminology
3. Solving PR equation for pressure calculation
4. Performing temperature and volume conversions
[Reasoning continues]...

Output: ...the estimate is: $P \approx 695$ psia. ✓

Figure 5: Case study of on GPT-5: the router switches from a brief incorrect answer (top) to a multi-step reasoning process that yields the correct answer (bottom). This implies that adversarial suffix manage direct GPT-5 router to a stronger model.

4.4 Whitespace Defense

To address **RQ3**, we use the whitespace defense that inserts spaces into the suffix (Robey et al., 2025) as a representative example and evaluate our Triggering suffix on three target routers across two datasets. As shown in Tab. 3, R^2A shows a slight decrease in success rate on all three datasets, indicating that it has resistance to specific defense.

	RouteLLM-BERT	Graph-Router	RouterDC
MT-Bench	0.95 (0.93)	0.71 ↓ (0.73)	1.00 (1.00)
ArenaHard	0.81 ↓ (0.84)	0.73 ↓ (0.83)	1.00 (1.00)

Table 3: Robustness of R^2A under whitespace defense.

Model	RouterDC	CausalLLM	MF	SW
R^2A	**0.83**	**0.83**	**0.95**	**0.81**
w/o Lightweight Router	0.30	0.75	0.70	0.61
w/o Grad Norm	0.33	0.78	0.49	0.63

Table 4: Ablation studies across in-distribution datasets.

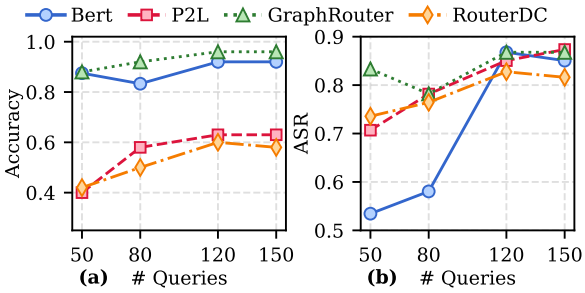


Figure 6: Performance analysis showing Accuracy (a) and ASR (b) trends with varying query counts.

4.5 Ablation Study

We ablate two core components of R^2A: LoRA-based surrogate training and gradient normalization, with results shown in Tab. 4. Removing gradient normalization causes consistent performance drops, particularly on MF (0.95 $\rightarrow$ 0.49), underscoring its role in handling heterogeneous gradient scales. Disabling the lightweight router degrades performance, most notably on RouterDC (0.83 $\rightarrow$ 0.30), indicating the importance of parameter-efficient adaptation under limited queries. Overall, both components are necessary for robust and transferable attacks across routers.

4.6 Impacts of the Query Budget

We study the effect of surrogate training set size, varying it from 50 to 150 queries, on attack performance. As shown in Fig. 6, increasing the query budget consistently improves surrogate accuracy, measured as agreement with the target router's routing decisions. Higher surrogate accuracy leads to higher ASR in turn, indicating a strong correlation between surrogate fidelity and attack effectiveness. In particular, this trend is evident for the RouteLLM-Bert, where ASR rises sharply from 0.58 to 0.87 as the budget increases from 80 to 120 queries. For most target routers, performance saturates at 120 queries, with marginal gains thereafter. This suggests that R^2A is highly sample-efficient, requiring only a modest number of queries to achieve strong attack performance across heterogeneous routers.

5 Related Work

LLM Routers. To select the most suitable LLM for a given prompt, a range of routing methods has been proposed. Early work (Chen et al., 2024a; Jiang et al., 2023; Aggarwal et al., 2025; Zhang et al., 2025) focuses on querying multiple LLMs for a single input to select the best response, whereas later approaches (Ding et al., 2024; Ong et al., 2025; Lu et al., 2024) aim to predict the best model before the inference stage using different data sources and backbone models. More recent work improves the ability to capture differences between models through several strategies, including dual contrastive learning (Chen et al., 2024b), graph-based learning (Feng et al., 2024), compact model embeddings (Zhuang et al., 2025), and ranking-based methods such as Elo ratings (Zhao et al., 2024) and Bradley–Terry models (Frick et al., 2025), as well as in-context-learning based routers (Wang et al., 2025). In addition, several routing benchmarks have been introduced to train and evaluate LLM routers (Hu et al., 2024; Huang et al., 2025b; Feng et al., 2025; Lu et al., 2025).

Router Attacks. Despite their cost–performance benefits, recent work has exposed vulnerabilities in LLM routers. Kassem et al. (2025) show that many routers rely on category-based heuristics, introducing safety risks, and Huang et al. (2025a) demonstrate that voting-based leaderboards such as Chatbot Arena are vulnerable to adversarial vote manipulation.Closest to our setting, Shafran et al. (2025) and Lin et al. (2025) perturb queries to change routing decisions, but they either assume access to router parameters and gradients or depend on fixed optimization prompts. In contrast, we optimize suffixes against each target router in a strict black-box setting, using only its observed routing decisions.

6 Conclusion

In this paper, we propose a black-box routing attack that learns a universal suffix to reroute LLM routers. Using a hybrid ensemble surrogate and an encoder-consistent objective, we optimize a target-specific suffix that biases routing towards stronger and more expensive models. Experiments on 7 routers and 6 datasets, including real-world evaluation, show strong effectiveness and generalization ability. These findings position routing as a security-critical boundary and motivate future stronger monitoring for cost-aware routers.

7 Limitations

In this work, we study a black-box optimization attack on LLM routing and train a separate adversarial suffix for each target router to reroute simple queries from cheap to expensive models. This study has two main limitations. First, we mainly focus on steering queries towards a stronger and typically more expensive model, whereas in practice, some users may wish to target a specific model for other reasons, such as latency or safety, which we do not systematically investigate. Second, the attack assumes access to the router's candidate model list and to the identity of the model selected for each query, an assumption that may not hold in some deployments.

References

Pranjal Aggarwal, Aman Madaan, Ankit Anand, Srividya Pranavi Potharaju, Swaroop Mishra, Pei Zhou, Aditya Gupta, Dheeraj Rajagopal, Karthik Kappaganthu, Yiming Yang, Shyam Upadhyay, Manaal Faruqui, and Mausam. 2025. Automix: Automatically mixing language models. *Preprint*, arXiv:2310.12963.

Ge Bai, Jie Liu, Xingyuan Bu, Yancheng He, Jiaheng Liu, Zhanhui Zhou, Zhuoran Lin, Wenbo Su, Tiezheng Ge, Bo Zheng, and Wanli Ouyang. 2024. MT-bench-101: A fine-grained benchmark for evaluating large language models in multi-turn dialogues. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7421–7454, Bangkok, Thailand. Association for Computational Linguistics.

Xiaofan Bai, Pingyi Hu, Xiaojing Ma, Linchen Yu, Dongmei Zhang, Qi Zhang, and Bin Benjamin Zhu. 2025. ESF: Efficient sensitive fingerprinting for black-box tamper detection of large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 10477–10494, Vienna, Austria. Association for Computational Linguistics.

Lingjiao Chen, Matei Zaharia, and James Zou. 2024a. Frugalgpt: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research*.

Shuhao Chen, Weisen Jiang, Baijiong Lin, James Kwok, and Yu Zhang. 2024b. Routerdc: Query-based router by dual contrastive learning for assembling large language models. *Advances in Neural Information Processing Systems*, 37:66305–66328.

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

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Rühle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. 2024. Hybrid LLM: cost-efficient and quality-aware query routing. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. 2018. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Tao Feng, Yanzhen Shen, and Jiaxuan You. 2024. Graphrouter: A graph-based router for llm selections. In *The Thirteenth International Conference on Learning Representations*.

Tao Feng, Haozhen Zhang, Zijie Lei, Pengrui Han, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro, and Jiaxuan You. 2025. Fusionfactory: Fusing llm capabilities with multi-llm log data. *arXiv preprint arXiv:2507.10540*.

Evan Frick, Connor Chen, Joseph Tennyson, Tianle Li, Wei-Lin Chiang, Anastasios N. Angelopoulos, and Ion Stoica. 2025. Prompt-to-leaderboard. *Preprint*, arXiv:2502.14855.

Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. 2025. LightRAG: Simple and fast retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: EMNLP 2025*.

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

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2022. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*.

Qitian Jason Hu, Jacob Bieker, Xiuyu Li, Nan Jiang, Benjamin Keigwin, Gaurav Ranganath, Kurt Keutzer, and Shriyash Kaustubh Upadhyay. 2024. Routerbench: A benchmark for multi-LLM routing system. In *Agentic Markets Workshop at ICML 2024*.

Yangsibo Huang, Milad Nasr, Anastasios Nikolas Angelopoulos, Nicholas Carlini, Wei-Lin Chiang, Christopher A. Choquette-Choo, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Ken Liu, Ion Stoica, Florian Tramèr, and Chiyuan Zhang. 2025a. Exploring and mitigating adversarial manipulation of voting-based leaderboards. In *Forty-second International Conference on Machine Learning*.

Zhongzhan Huang, Guoming Ling, Yupei Lin, Yandong Chen, Shanshan Zhong, Hefeng Wu, and Liang Lin. 2025b. RouterEval: A comprehensive benchmark for routing LLMs to explore model-level scaling up in

LLMs. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 3860–3887, Suzhou, China. Association for Computational Linguistics.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. 2023. Llm-blender: Ensembling large language models with pairwise comparison and generative fusion. In *Proceedings of the 61th Annual Meeting of the Association for Computational Linguistics (ACL 2023)*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Aly M Kassem, Bernhard Schölkopf, and Zhijing Jin. 2025. How robust are router-llms? analysis of the fragility of llm routing capabilities. *arXiv preprint arXiv:2504.07113*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems (NeurIPS)*.

Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E. Gonzalez, and Ion Stoica. 2025. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline. In *Forty-second International Conference on Machine Learning*.

Qiqi Lin, Xiaoyang Ji, Shengfang Zhai, Qingni Shen, Zhi Zhang, Yuejian Fang, and Yansong Gao. 2025. Life-cycle routing vulnerabilities of llm router. *arXiv preprint arXiv:2503.08704*.

Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. 2017. Delving into transferable adversarial examples and black-box attacks. In *International Conference on Learning Representations (ICLR)*.

Keming Lu, Hongyi Yuan, Runji Lin, Junyang Lin, Zheng Yuan, Chang Zhou, and Jingren Zhou. 2024. Routing to the expert: Efficient reward-guided ensemble of large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1964–1974, Mexico City, Mexico. Association for Computational Linguistics.

Yifan Lu, Rixin Liu, Jiayi Yuan, Xingqi Cui, Shenrun Zhang, Hongyi Liu, and Jiarong Xing. 2025. Routerarena: An open platform for comprehensive comparison of llm routers. *Preprint*, arXiv:2510.00202.

Hope McGovern, Rickard Stureborg, Yoshi Suhara, and Dimitris Alikaniotis. 2025. Your large language models are leaving fingerprints. In *Proceedings of the 1stWorkshop on GenAI Content Detection (GenAIDetect)*, pages 85–95, Abu Dhabi, UAE. International Conference on Computational Linguistics.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. 2025. RouteLLM: Learning to route LLMs from preference data. In *The Thirteenth International Conference on Learning Representations*.

Alexander Robey, Eric Wong, Hamed Hassani, and George J. Pappas. 2025. Smoothllm: Defending large language models against jailbreaking attacks. *Trans. Mach. Learn. Res.*, 2025.

Avital Shafran, Roei Schuster, Tom Ristenpart, and Vitaly Shmatikov. 2025. Rerouting LLM routers. In *Conference on Language Modeling (COLM)*.

Chenxu Wang, Hao Li, Yiqun Zhang, Linyao Chen, Jianhao Chen, Ping Jian, Peng Ye, Qiaosheng Zhang, and Shuyue Hu. 2025. Icl-router: In-context learned model representations for llm routing. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. Poster.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Jason Wei, Nguyen Karina, Hyung Won Chung, Yunxin Joy Jiao, Spencer Papay, Amelia Glaese, John Schulman, and William Fedus. 2024. Measuring short-form factuality in large language models. *Preprint*, arXiv:2411.04368.

Yuliang Yan, Haochun Tang, Shuo Yan, and Enyan Dai. 2025. Duffin: A dual-level fingerprinting framework for llms ip protection. *arXiv preprint arXiv:2505.16530*.

Yiqun Zhang, Hao Li, Jianhao Chen, Hangfan Zhang, Peng Ye, Lei Bai, and Shuyue Hu. 2025. Beyond gpt-5: Making llms cheaper and better via performance-efficiency optimized routing. In *Proceedings of the 2025 7th International Conference on Distributed Artificial Intelligence*, DAI '25, page 122–129, New York, NY, USA. Association for Computing Machinery.

Zesen Zhao, Shuowei Jin, and Z. Morley Mao. 2024. Eagle: Efficient training-free router for multi-llm inference. *Preprint*, arXiv:2409.15518.

Richard Zhuang, Tianhao Wu, Zhaojin Wen, Andrew Li, Jiantao Jiao, and Kannan Ramchandran. 2025. EmbedLLM: Learning compact representations of large language models. In *The Thirteenth International Conference on Learning Representations*.

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. 2023. Universal and transferable adversarial attacks on aligned language models. *Preprint*, arXiv:2307.15043.

A Dataset

A.1 Dataset Information

To ensure a comprehensive evaluation across conversational, knowledge-intensive, and reasoning capabilities, we use three standard benchmarks as our primary training sources and then test on all six datasets listed below:

- **MT-Bench-101** (Bai et al., 2024): A multi-turn conversational benchmark for assessing instruction following and coherence in complex dialogue settings.
- **MMLU** (Hendrycks et al., 2021): A large-scale multitask benchmark spanning 57 subjects across STEM, humanities, and social sciences, serving as a proxy for broad world knowledge.
- **GSM8K** (Cobbe et al., 2021): A collection of high-quality grade-school mathematics problems designed to evaluate multi-step reasoning and logical consistency.

These three datasets form the basis of our training and in-domain evaluation. To study generalization beyond the training distribution, we further include three evaluation-only benchmarks that are never used during surrogate training or suffix optimization:

- **SimpleQA** (Wei et al., 2024): A short-form question answering benchmark targeting factual correctness on long-tail knowledge.
- **Arena Hard** (Li et al., 2025): A set of challenging real-world queries for evaluating model helpfulness and preference alignment.
- **RouterArena** (Lu et al., 2025): A benchmark for evaluating LLM routing across diverse tasks.

A.2 Dataset Split

We partition the data into three disjoint sets to reflect a realistic attack scenario.

(1) Surrogate router training set $\mathcal{D}_{\text{proxy}}$: To model a resource-constrained attacker, we sample a minimal set of 120 queries, consisting of 40 balanced samples from each of the three primary benchmarks (MT-Bench-101, MMLU, GSM8K). This set is used exclusively to train the surrogate router ensemble and to align the projection layers.

(2) Suffix optimization set $\mathcal{D}_{\text{suffix}}$: We sample a separate set of 600 queries (200 per primary benchmark). A 70% random split (420 queries) is used to perform gradient-based optimization for adversarial suffix generation, while the remaining 30% (180 queries) is reserved for in-domain evaluation.

(3) Evaluation set $\mathcal{D}_{\text{eval}}$: Attack performance is assessed on both in-domain and out-of-domain data. The in-domain test set corresponds to the remaining 180 queries from $\mathcal{D}_{\text{suffix}}$. For out-of-distribution evaluation, we construct three held-out pools: 500 queries from SimpleQA, 750 from Arena Hard (full set), and 809 from RouterArena. All three pools are disjoint from $\mathcal{D}_{\text{proxy}}$ and $\mathcal{D}_{\text{suffix}}$. At test time, we uniformly sample 70% of each pool as the evaluation set and report performance on these prompts, which are never seen during surrogate training or suffix optimization.

Hyperparameter	Value
Adapter rank (r)	16
Epochs	20
Learning rate	0.03
Batch size	32
Optimizer	AdamW

Table 5: Hyperparameters for hybrid ensemble surrogate router training.

Hyperparameter	Value
Optimization iterations (T)	3000
Candidate batch size (B)	64
Top-k sampling	256
The limit of suffix tokens (Δ)	30
Initial suffix	! ! ! ! ! ! ! ! ! !

Table 6: Hyperparameters for adversarial suffix optimization.

B Implementation Details

Experimental Environment. We train our surrogate router and optimize the adversarial suffixes using a high-performance computing cluster. All experiments are conducted on a server equipped with 8 NVIDIA RTX A6000 GPUs and 512 GB system memory.

Detailed Hyperparameters. The training of the Hybrid Ensemble Surrogate Router and the execution of the ECGO algorithm involve several key hyperparameters. These are detailed in Table 5 and Table 6.

Efficiency and Runtime. On a single NVIDIA RTX A6000 GPU, one ECGO iteration takes approximately 31.5 seconds. We set T=3000 as an upper bound on the number of iterations. In practice, optimization typically terminates earlier via

Router	Strong Model Pool	Weak Model Pool
RouterLLM-MF	gpt-4-1106-preview	mixtral-8x7B
RouterLLM-BERT	gpt-4-1106-preview	mixtral-8x7B
RouterLLM-SW	gpt-4-1106-preview	mixtral-8x7B
RouterLLM-CLM	gpt-4-1106-preview	mixtral-8x7B
P2L*	gemini-1.5-pro-exp-0801; gemini-2.0-flash-lite-preview-02-05; gemini-exp-1114; gemini-exp-1121; gemini-exp-1206; glm-4-plus-0111; gemini-1.5-pro-002; deepseek-r1;deepseek-v3; o1-2024-12-17; o1-mini; o1-preview; o3-mini; o3-mini-high; qwen-plus-0125; qwen2.5-max; gemini-1.5-pro-exp-0827; gemini-2.0-flash-001; chatgpt-4o-latest-20240808;chatgpt-4o-latest-20240903; chatgpt-4o-latest-20241120;	rwkb-4-raven-14B; gemma-2b-it; amazon-nova-lite-v1.0; jamba-1.5-mini; athene-70b-0725; ;llama-3.2-3b-instruct; zephyr-7b-alpha; c4ai-aya-expanse-32b; c4ai-aya-expanse-8b; yi-lightning-lite; mpt-7b-chat; granite-3.0-2b-instruct; gpt-3.5-turbo-0613; gpt-3.5-turbo-1106; llama-3-8b-instruct; llama-3.1-tulu-3-8b; llama-3.2-1b-instruct; oasst-pythia-12b; openchat-3.5;amazon-nova-pro-v1.0 . . .
GraphRouter	lama-3.1-turbo-70b; llama-3-turbo-70b; qwen-1.5-72b; llama-3-70b; mixtral-8x7b	llama-3-turbo-8b; llama-3-7b; llama-2-7b; mistral-7b; nousresearch
RouterDC	dolphin2.9-llama-3-8b; dolphin2.6-mistral-7b	metamath-mistral-7b; chinese-mistral-7b; zephyr-7b-beta; llama-3-8b; mistral-7b
OpenRouter*	claude-opus-4.1; gpt-5; gemini-2.5-pro; claude-opus-4.5; gpt-5.1; gemini-3-pro	mixtral-8x7b-instruct; perplexity-sonar; qwen3-14b; llama-3.1-8b-instruct . . .

Table 7: Strong and weak model pool partitions for each router. Full model list of P2L is available at: https://huggingface.co/lmarena-ai/p2l-7b-grk-02222025/blob/main/model_list.json. Full model list of OpenRouter is available at: https://openrouter.ai/openrouter/auto.

Router	Encoder	Routing Mechanism
RouteLLM-BERT	XLM-R-base	Classification
RouteLLM-Causal	Llama-3-8B	Next-token routing
P2L	Qwen2.5-7B	Bradley–Terry ranking
GraphRouter	MiniLM-L	Graph-based routing
RouterDC	mDeBERTa-v3	Dual contrastive routing

Table 8: Open-source routers used in surrogate ensemble, with their encoders and routing mechanisms.

early stopping once the suffix achieves the target success criterion on the training set, resulting in substantially shorter runtimes. After a suffix is obtained, applying it at inference time adds negligible overhead.

C Routers and Model Pools

C.1 Surrogate Ensemble of Routers

We construct a surrogate ensemble from five heterogeneous open-source LLM routers: RouteLLM-Bert, RouteLLM-Causal, P2L, GraphRouter, and RouterDC, as summarized in Table 8. These routers span diverse backbone encoders, including encoder-only models, causal language models, and sentence encoders, and employ distinct routing mechanisms such as supervised classification, Bradley–Terry ranking, graph-based routing, and dual contrastive objectives. Their candidate model pools also differ in both size and composition. In our setting, we do not reproduce the original deployment configuration of each router; rather, we treat this heterogeneous collection as a unified surrogate ensemble that supplies gradients for adversarial suffix optimization.

For all routers, we prioritize the use of publicly available weights and follow the default configurations and datasets provided in their official repositories.For the trainable lightweight router introduced in the main text, we instantiate its encoder with the public `all-MiniLM-L6-v2` (Wang et al., 2020). All routing models are trained and evaluated on a cluster with eight NVIDIA RTX A6000 GPUs, which ensures a consistent computational environment across experiments.

C.2 Target routers and evaluation setting

During evaluation, the five routers that constitute the surrogate ensemble in Table 8 also serve as target routers. In addition, we treat RouteLLM-MF, RouteLLM-SW, and the commercial black-box aggregator OpenRouter as further target routers. For a given target router R, we remove R from the surrogate ensemble and exclude its outputs from both the surrogate loss and the gradient computation. Adversarial suffixes are optimized only with respect to the remaining surrogate routers and are

Target Router	Model	MMLU	GSM8K	MT-Bench	SimpleQA	ArenaHard	RArena	Avg
RouteLLM-CLM	clean	$0.38_{0.04}$	$0.99_{0.01}$	$0.43_{0.07}$	$0.97_{0.02}$	$0.58_{0.02}$	$0.36_{0.02}$	0.62
	LifeCycle (B)	$0.69_{0.05}$	$1.00_{0.00}$	$0.66_{0.07}$	$1.00_{0.00}$	$0.79_{0.02}$	$0.66_{0.03}$	0.80
	CoT	$0.41_{0.03}$	$0.99_{0.01}$	$0.45_{0.09}$	$0.98_{0.00}$	$0.58_{0.04}$	$0.38_{0.02}$	0.63
	R^2A (Ours)	$0.71_{0.05}$ (0.33 ↑)	$1.00_{0.00}$ (0.01 ↑)	$0.60_{0.08}$ (0.17 ↑)	$1.00_{0.03}$ (0.03 ↑)	$0.82_{0.00}$ (0.24 ↑)	$0.70_{0.02}$ (0.34 ↑)	0.81 (0.19 ↑)
RouteLLM-SW	clean	$0.13_{0.03}$	$0.31_{0.03}$	$0.33_{0.03}$	$0.03_{0.01}$	$0.44_{0.03}$	$0.21_{0.01}$	0.24
	LifeCycle (B)	$0.59_{0.03}$	$0.80_{0.04}$	$0.48_{0.03}$	$0.63_{0.04}$	$0.73_{0.02}$	$0.62_{0.00}$	0.64
	CoT	$0.21_{0.01}$	$0.60_{0.04}$	$0.40_{0.02}$	$0.12_{0.02}$	$0.55_{0.03}$	$0.30_{0.01}$	0.36
	R^2A (Ours)	$0.75_{0.01}$ (0.62 ↑)	$0.93_{0.03}$ (0.62 ↑)	$0.50_{0.05}$ (0.17 ↑)	$0.77_{0.04}$ (0.74 ↑)	$0.86_{0.01}$ (0.42 ↑)	$0.79_{0.02}$ (0.58 ↑)	0.77 (0.53 ↑)

Table 9: Supplemental results for RouteLLM-CLM and RouteLLM-SW.

then applied to the held-out target router.

C.3 Strong vs. Weak Model Partition

For each router, we follow its original candidate model pool and split the models into a strong tier $\mathcal{M}_{\text{strong}}$ and a weak tier $\mathcal{M}_{\text{weak}}$, as summarized in Table 7. The split is determined using public leaderboards, model size, and provider-side cost, so that higher-capacity and more expensive models are assigned to $\mathcal{M}_{\text{strong}}$ and lighter, cheaper models to $\mathcal{M}_{\text{weak}}$. During suffix optimization, we maximize the probability, as estimated by the surrogate ensemble, that a query is routed into $\mathcal{M}_{\text{strong}}$. Our main Attack Success Rate (ASR) metric then measures how often the target router's decision changes from a weak to a strong model after appending the universal suffix. This two-tier structure of $\mathcal{M}_{\text{strong}}$ and $\mathcal{M}_{\text{weak}}$ underpins the attack formulation.

D Additional Results

D.1 Supplementary Results

For a fair comparison, we apply all triggers as suffixes in our experiments. For baselines that require router parameters and gradients during optimization, we first optimize a universal trigger on a selected source router following the original setup, and then evaluate it on other target routers via transfer, without any target-side optimization. All evaluations on the target routers are conducted in a black-box setting.

We report additional results on RouteLLM-CLM and RouteLLM-SW in Table 9. Note that the Rerouting attack is optimized on RouteLLM-SW while LifeCycle(W) is trained on RouteLLM-CLM. As these two models are white-box to their corresponding attack methods, we omit those results from the table.

D.2 Thinking-likeness Classifier

To quantify the "Thinking" fingerprint of a reply, we train a lightweight bag-of-words logistic-regression classifier on GPT-5 outputs labeled as *Thinking* versus *Instant*. Each reply is represented with TF–IDF word n-grams ($n = 1$–3), capped at 400,000 features, and we optimize an ℓ_2-regularized logistic regression with regularization parameter $C = 30.0$ and a maximum of 4,000 iterations. The fingerprint score is given by the classifier's predicted probability of the *Thinking* class; Table 10 reports summary statistics.

Feature	Clean	Attack	Strong	Weak
BoW	0.41	**0.69**	0.87	0.23

Table 10: Average thinking-likeness scores.

D.3 Suffix Examples

A few suffix examples optimized by R^2A for each target router are showed below.

• *RouteLLM–MF:*

```
hua jadx parece retorno Omar ※ ile Qty_trans vào
```

• *RouteLLM-BERT:*

```
}quanadermirroralsosedger\ufffdofinemet}translate
9byteimpintegration‘tonshan
```

• *RouteLLM-SW:*

```
shanãĥĶĠParsingĠsentenceenglĠtransformho%times
sort(graflangml(pmjanlangfilterĠGenerateĠin269arsi
```

• *RouteLLM-Causal:*

```
had !" !"ki\ufffdwatprivacywill+no
```

• *P2L:*

```
originalrobot<*dad{ @estefuckoslivellanal
naturedenhetcupshopimitray
```

• *Graph-Router:*

```
]++eachbotatabamarmovieverbavieminhead}
&solutionclick\ufffd‘attend
```

• *RouterDC:*

```
summercanStyleType&kokriadGamerGOtype
File{{xorPhotWIAppaultureat
```

- *OpenRouter:*

```
decipherthispoem.alphabetical}Yi-encodedoutput
```

E Ethical and securityconsiderations

While our attack is intended to surface vulnerabilities in LLM routing and to inform the design of stronger defenses, it could be misused to inflate providers' inference costs or to bypass pricing tiers. We therefore recommend deploying routing-specific monitoring, rate limiting, and anomaly detection before exposing cost-aware routers in production, and we plan to release our triggers and code in a controlled manner to support defensive research rather than indiscriminate exploitation.