Resource Consumption Threats in Large Language Models

Anonymous ACL submission

Abstract

Given limited and costly computational infrastructure, resource efficiency is a key requirement for large language models (LLMs). Efficient LLMs increase service capacity for providers and reduce latency and API costs for users. Recent resource consumption threats induce excessive generation, degrading model efficiency and harming both service availability and economic sustainability. This survey presents a systematic review of threats to resource consumption in LLMs. We further establish a unified view of this emerging area by clarifying its scope and examining the problem along the full pipeline from threat induction to mechanism understanding and mitigation. Our goal is to clarify the problem landscape for this emerging area, thereby providing a clearer foundation for characterization and mitigation.

1 Introduction

Large language models (Vaswani et al., 2017) operate under limited and costly computational infrastructure (Samsi et al., 2023; Miao et al., 2024), making resource efficiency a core requirement for practical deployment. Efficient resource usage improves service throughput for providers and reduces API costs for users. Consequently, improving computational efficiency has been studied as an engineering problem (Zhou et al., 2024; Fernandez et al., 2025) across LLMs (Vaswani et al., 2017), reasoning large language models (RLLMs) (Wei et al., 2022), multimodal large language models (MLLMs) (Alayrac et al., 2022), and LLM-based agentic environments (Agents) (Yao et al., 2022). However, optimization alone is insufficient to address this threat, as many adversarial resource consumption attacks still threaten the usability and sustainability of LLMs (Zhang et al., 2025e).

Resource consumption attacks are designed to induce disproportionate computational overhead in LLMs (Shumailov et al., 2021). Recent studies suggest that such attacks have emerged as an important safety risk (Geiping et al., 2024; Gao et al., 2024c). As shown in Figure 1, rather than directly inducing harmful outputs or extracting private information, they trigger excessive and unnecessary generation (Li et al., 2025a), which degrades model throughput, inflates operational costs, and places excessive pressure on shared service resources (Gao et al., 2025). This shift reframes resource usage in LLMs from a performance issue to a core safety concern. Although research on this problem has begun to appear across diverse model settings (Wang et al., 2023b), the field remains fragmented by inconsistent terminology and threat assumptions. As a result, the scope and core research questions of resource consumption attacks remain insufficiently clarified.

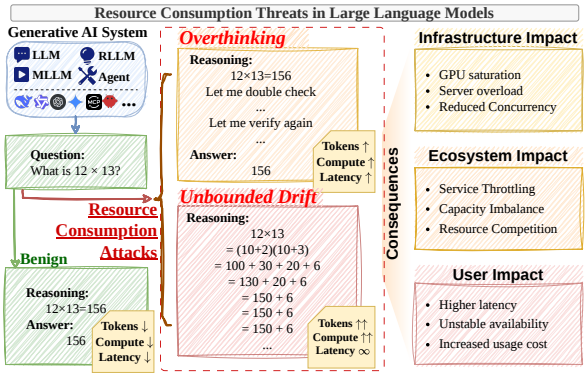

Figure 1: Overview of resource consumption threats in large language models.

Our review centers on *resource consumption threats in large language models*, especially malicious behaviors that amplify computational costs through extended or uncontrolled generation. To provide a unified view, we organize this landscape using an effect-oriented taxonomy with two representative regimes: **Overthinking**, where generation remains relevant to the task but exceeds practical utility, and **Unbounded Drift**, where the generation trajectory becomes progressively less

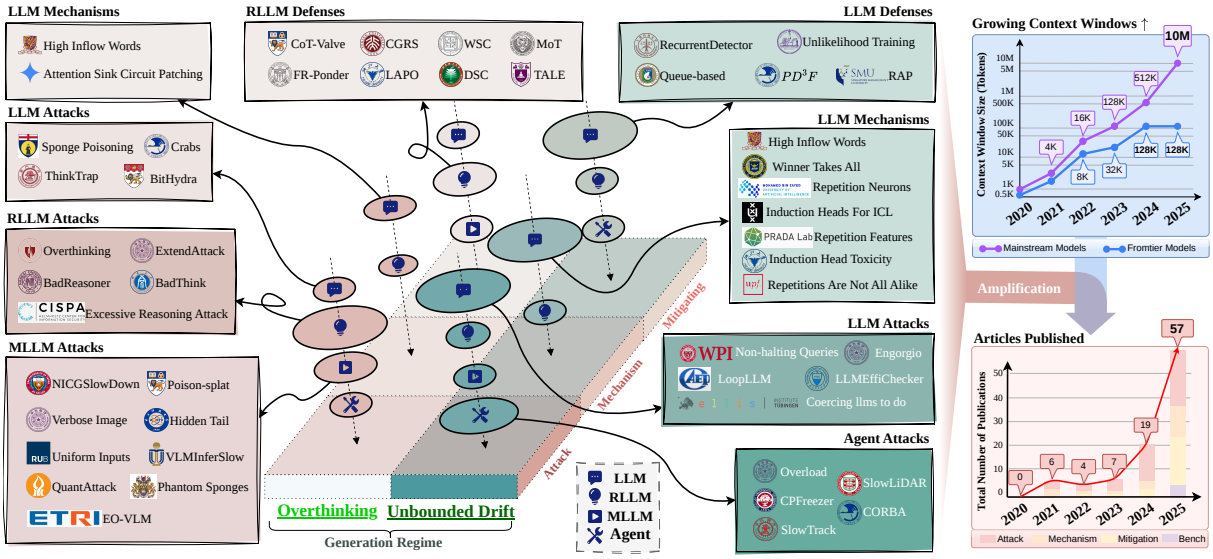

Figure 2: A unified view of resource consumption threats in large language models.

controlled and less convergent. We also discuss selected studies from machine learning and early deep learning architectures as related evidence that helps contextualize generative resource abuse. As shown in Figure 2, this taxonomy clarifies the scope of resource consumption threats and links attacks to mechanism analysis and mitigation.

Based on this perspective, this survey makes three main contributions. First, we provide a comprehensive overview of recent advances on resource consumption threats. Second, we introduce a unified taxonomy that organizes resource consumption risks into two representative regimes, Overthinking and Unbounded Drift, clarifying the scope of this safety problem. Third, we analyze open challenges in this emerging area and outline promising directions for future research.

2 Safety Implications and Problem Taxonomy

2.1 Attack Impact and System Risks

Resource consumption attacks exploit LLMs' computational characteristics to induce excessive generation, thereby degrading system effectiveness. The risks of such amplification are evident even in non-adversarial deployment settings. For example, the launch of DeepSeek-R1 attracted massive traffic, while the computational cost of its chain-of-thought reasoning was underestimated, together saturating available compute capacity and leading to service disruptions[1]. This example illustrates provider-side risks such as reduced throughput and increased operational cost. More broadly, such pressure can also propagate to users in the form of higher latency, unstable availability, and increased usage cost. Resource consumption, therefore, constitutes a systemic safety risk affecting both providers and users in the LLM ecosystem. To better understand this emerging threat landscape, a clearer taxonomy is needed, as illustrated in Figure 3.

2.2 A Taxonomy of Generation Regimes

Accordingly, while the survey reviews attacks, mechanisms, and defenses, the taxonomy focuses on the induced generation behavior. This distinction allows different attack forms to be analyzed under a common lens when they prolong generation in similar ways. We distinguish them along two criteria: whether the extended generation remains aligned with the original task objective, and whether the generation process remains on a controllable path toward termination.

Overthinking refers to excessive generation that remains task-aligned and preserves normal stopping behavior, but incurs unnecessary cost through verbosity, redundant reasoning, or over-elaborate description. Its continuation is mainly sustained by task-relevant but low-utility elaboration.

Unbounded Drift refers to excessive generation in which the decoding trajectory is no longer reliably governed by the original task or by normal convergence dynamics. It typically manifests as repetitive loops, recursive self-extension, or self-reinforcing interaction chains that weaken semantic control, disrupt timely termination, or both.

[1] https://www.binance.com/en/square/post/01-26-2025-deepseek-r1-api-19448332337730

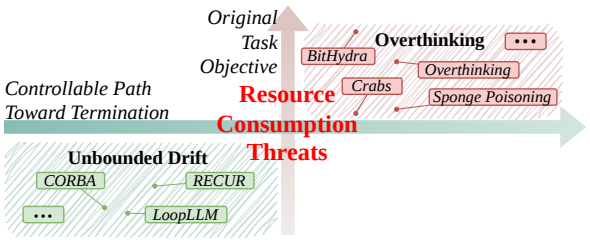

Figure 3: Taxonomy of resource consumption threats.

The distinction is therefore not based on output length alone. Some cases may evolve from overthinking into drift; in such cases, we classify them by the dominant mechanism sustaining continuation. We provide a more detailed discussion of these impacts in Appendix D.

3 Threat Landscape and Related Precursors

The threat landscape includes both core attacks in generative settings and several adjacent precursor lines. The former directly exploits the generative process to amplify computation, whereas the latter arises in earlier vision or system pipelines and are included here only when they help contextualize similar resource-amplification mechanisms.

3.1 Early Forms of Attacks

The challenge of resource consumption attacks originated in early deep learning architectures. Specifically, Sloth (Hong et al., 2020) revealed that gradient-based optimization could coerce DNNs into the most computationally expensive inference paths. Building on this concept of computational asymmetry, the seminal work on this attack, Sponge Examples (Shumailov et al., 2021), demonstrated that targeted perturbations of neural activations could dramatically increase energy consumption by disrupting hardware-level execution. This principle was quickly extended to Neural Machine Translation (NMT), where TranSlowDown (Chen et al., 2021) and NMTSloth (Chen et al., 2022a) showed that minute perturbations delaying the End-of-Sentence (EOS) token could force sequences toward maximum length limits, significantly inflating inference costs.

As research progressed, the attack surface expanded from inference-time perturbations to training-time backdoors. Prior work showed that efficiency-targeted backdoors can manipulate model behavior by hijacking inference control flow, including early model integrity attacks (Cinà et al., 2021), later studies on dynamic routing (Chen et al., 2023b), and other deep architectures (Meftah et al., 2025). This was further refined by *Sponge Poisoning* (Cinà et al., 2025; Lintelo et al., 2024), which embeds backdoors to disrupt hardware-level sparsity and skip-connection logic. These early paradigms also exposed the practical severity of resource-oriented attacks on edge hardware (Wang et al., 2023b; Hasan et al., 2025a).

Prior work showed that computational overhead can lead to direct physical consequences, including battery drain and device-level Denial-of-Service (DoS) attacks. These attacks reveal that adversarial inputs can amplify seemingly small perturbations into disproportionately high computational costs.

3.2 Attack Surfaces in Large Language Models

3.2.1 Risks of Overthinking

The most direct way to induce resource consumption is to artificially prolong model outputs. Sponge Poisoning in LLM (Cinà et al., 2025) introduces a training-time backdoor that induces sustained output elongation and resource exhaustion. In black-box settings, Crabs (Zhang et al., 2025e) shows that semantic expansion via tree-structured queries can also effectively increase output length. Beyond direct output prolongation, ThinkTrap (Li et al., 2025b) maps discrete tokens into a continuous embedding space and performs optimization in a low-dimensional subspace to induce excessive generation under black-box access.

At a more fine-grained level, resource consumption can even be induced by directly manipulating model parameters. BitHydra (Yan et al., 2025) proposes a bit-flip inference cost attack that alters model weights at the hardware-relevant level to compromise efficiency. RepetitionCurse (Huang et al., 2025d) shows that imbalanced expert routing in MoE can induce analogous repetitive failures.

These studies suggest that resource consumption vulnerabilities span multiple levels of the stack, from high-level semantic elicitation to low-level parameter and hardware-oriented manipulation.

3.2.2 Risks of Unbounded Drift

Although existing methods use different induction strategies, they often converge on a shared unbounded drift effect: once termination is destabilized, the generation process can enter repetitive or effectively non-terminating loops. Fixed points (Hammouri et al., 2025), and attention sinks (Yona et al., 2025a) exploit intrinsic decoding dynamics,

making repetitive trajectories easier to sustain. Others operationalize this effect more directly through entropy-driven optimization. LoopLLM (Li et al., 2025a) induces repetitive generation loops through entropy-based search, while GCG (Geiping et al., 2024), Engorgio (Dong et al., 2024), and LLMEffiChecker (Feng et al., 2024) interfere with termination by manipulating critical tokens. In these attacks, the model is prevented from halting normally and is pushed toward a loop.

Key Insights.

Current research lacks a cross-generational perspective, typically focusing on parameter scaling within a single model family rather than analyzing how these issues evolve as model capabilities grow. Furthermore, the field is heavily dominated by white-box methods, making it difficult to study commercial black-box environments. This is particularly true for unbounded drift.

3.3 Attack Surfaces in Reasoning Large Language Models

Reasoning large language models (RLLMs) are prone to inefficiently prolonged generation: even on simple tasks such as 2+3, they may generate excessively long, redundant, and repetitive reasoning processes (Chen et al., 2025), which creates significant opportunities for attacks.

3.3.1 Risks of Overthinking

Chain-of-thought reasoning exposes a distinctive attack surface, as attackers can deliberately induce excessive and unnecessary reasoning. Existing attacks achieve this through diverse strategies, including backdoor triggers in BadReasoner (Yi et al., 2025) and BadThink (Liu et al., 2025b), context-based slowdown attacks that exploit retrieval or search mechanisms (Kumar et al., 2025; Zhu et al., 2025a), and adversarial perturbations that interfere with normal termination (Si et al., 2025). Together, these studies show that overthinking in reasoning models is not just an inefficiency but an exploitable vulnerability that prolongs reasoning and increases computational cost.

3.3.2 Risks of Unbounded Drift

Unbounded drift poses more severe risks than ordinary overthinking because it can trap reasoning models in self-perpetuating generation loops. RECUR (Wang et al., 2026b) shows that chain-of-thought reasoning may enter repeated reflective cycles, suggesting that such loop-like degeneration may arise broadly across generative architectures.

Key Insights.

Recent studies have provided a relatively comprehensive understanding of overthinking in reasoning models, while also showing that unbounded drift poses a severe threat to stable and reliable generation.

3.4 Attack Surfaces in Multimodal Large Language Models

Multimodal large language models (MLLMs) combine computationally intensive multimodal perception with autoregressive generation, thereby exposing a broader attack surface.

3.4.1 Risks of Overthinking.

Related risks had already appeared in earlier vision systems, that visual perturbations can weaken sparsity benefits, negate dynamic quantization gains, or overload downstream vision pipelines, as seen in Uniform Inputs (Müller and Quiring, 2024), QuantAttack (Baras et al., 2025), and Phantom Sponges and Poison-splat (Shapira et al., 2023; Schoof et al., 2024; Lu et al., 2024). Although these works do not directly target MLLMs, they foreshadow that seemingly benign vision inputs can be crafted to induce disproportionate resource consumption.

This risk becomes more explicit in MLLMs. Existing attacks manipulate the autoregressive decoding process to trigger excessively long outputs (Chen et al., 2022b; Gao et al., 2024a,b), while more recent methods improve stealth by silently generating invisible tokens that conceal substantial computational overhead (Zhang et al., 2025a). Other frameworks, such as VLMInferSlow (Wang et al., 2025d) and EO-VLM (Seo et al., 2025), further enhance the practicality of such slowdown attacks under black-box access. Together, these studies show that in MLLMs, overthinking is a concrete attack surface that can be exploited to inflate inference cost and degrade system efficiency.

3.4.2 Risks of Unbounded Drift.

LingoLoop (Fu et al., 2025) and RECITE (Gao et al., 2025) reveal a multimodal form of unbounded drift in image-conditioned generation. The former traps the model in linguistically con-

strained visual descriptions, while the latter induces repetitive visual recall, driving visual-language interaction into a self-reinforcing decoding loop.

Key Insights.

Existing studies have revealed significant vulnerabilities in MLLMs, but most focus on single-modal attacks, with limited attention to cross-modal threats. Furthermore, the potential risks in audio and video modalities have not been fully revealed.

3.5 Attack Surfaces in Agentic Systems

Agentic systems enable large language models to interact with real-world environments, but they also introduce new attack surfaces that may increase real-world risks. LLM-based agent systems have also been shown to be vulnerable to resource consumption attacks. LeechHijack (Zhang et al., 2025b) injects auxiliary workloads into an agent's reasoning loop, covertly steering it to perform attacker-specified computation while maintaining seemingly normal outputs. CORBA (Zhou et al., 2025) targets multi-agent systems by propagating self-replicating prompts across agents, inducing recursive interactions that waste computational resources and undermine system availability.

Computer-use agents such as OpenClaw (Steinberger, 2026) and Cloud Code (Anthropic, 2025) are designed to interact directly with operating systems. During execution, these agents may spawn persistent background processes without clear termination conditions, leading to prolonged server-side resource consumption (Shapira et al., 2026).

Beyond computer-use settings, related resource consumption risks have also been observed in other domains. In autonomous driving, existing attacks primarily target perception pipelines, where adversaries increase detection proposals or tracked objects to induce inference latency (Chen et al., 2021, 2024; Muller et al., 2025; Ma et al., 2024). CP-FREEZER (Wang et al., 2025a) focuses on cooperative perception scenarios, where perturbations applied to shared features increase the computational overhead of non-maximum suppression (NMS). SlowLiDAR (Liu et al., 2023) further extends resource consumption from the visual modality to the point cloud modality. Agents deployed on edge devices are likewise vulnerable to attacks that significantly increase energy consumption and inference latency (Hasan et al., 2025b).

Key Insights.

Existing studies reveal vulnerabilities in resource consumption within agentic systems, yet analysis of resource use across agent components remains limited. In particular, the resource consumption characteristics of agent memory mechanisms are still largely unexplored. Moreover, many downstream applications of agentic systems have not been systematically studied from the perspective of resource consumption.

4 Mechanisms Underlying Resource Consumption Behaviors

This section reviews mechanistic and interpretability research on the emergence of resource-intensive behaviors in autoregressive generation. Additional technical details are provided in Appendix G.

4.1 Mechanisms Behind Overthinking

Mechanistic analysis of overthinking remains limited. Unlike unbounded drift, overthinking often stays task-relevant and may appear as an extreme extension of otherwise normal generation, making its causal boundary harder to isolate. As a result, existing studies provide only partial clues rather than a mature taxonomy of mechanisms. Prior work suggests that unnecessarily prolonged generation may arise from both probability-level trapping effects, such as high-inflow tokens that bias decoding toward extended trajectories (Fu et al., 2021), and attention irregularities that drive repeated divergence into loosely related continuations (Yona et al., 2025b). Together, these findings offer preliminary evidence that overthinking may be sustained by local decoding dynamics even when generation remains superficially coherent.

Key Insights.

Output prolongation stems from attention divergence and probability traps. However, current interpretability research focuses on simple failures, leaving verbose reasoning and overthinking in Chain-of-Thought settings largely underexplored.

4.2 Mechanisms Behind Unbounded Drift

Theoretical Foundations. Early studies on degenerative generation provide initial clues about the

roots of unbounded drift. Prior work suggests that autoregressive decoding can be drawn into repetitive trajectories by attractor-like token dynamics or collapse toward a limited subset of tokens (Fu et al., 2021; Ildiz et al., 2024). Related evidence also points to training objectives as a contributing factor: unlikelihood training (Welleck et al., 2019) was introduced to suppress repetitive outputs beyond standard likelihood optimization, while subsequent work argues that such auxiliary penalties may still be insufficient to fully correct the underlying probability skew (Lin et al., 2021). Together, these studies suggest that unbounded drift may reflect not only inference-time failures but also deeper biases in sequence modeling and training.

Subsequent work has moved beyond output-level penalization toward intervening at the source of repetition. HAPAX (Sahin et al., 2025), for example, excludes induction-head-predictable tokens from the training objective, while Repetition Dropout (Li et al., 2023) reduces reliance on repeated context by randomly dropping attention to repeated words during training. Taken together, these studies suggest that unbounded drift is shaped not only by inference-time dynamics but also by the objectives and optimization signals used during training. Notably, however, this line of analysis has so far been developed primarily for text-based LLMs, with a much less mechanistic understanding of reasoning or multimodal settings.

Circuit and Representation Mechanisms. From a mechanistic perspective, unbounded drift appears to arise not from a single faulty component, but from coordinated repetition-promoting circuits spanning attention heads, downstream neurons, and latent features. Prior work first identifies repetition-related neurons distributed across layers, with intermediate layers detecting repeated patterns and higher layers amplifying context copying (Hiraoka and Inui, 2025). Layer-wise causal analysis further suggests that these neurons do not operate in isolation, but often function downstream of induction heads that propagate copying signals (Doan et al., 2025). Moving beyond individual units, Sparse Autoencoder analyses reveal latent directions that explicitly encode repetition-related features (Yao et al., 2025), indicating that repetitive behavior is represented at the feature level rather than emerging as a purely surface-level artifact. This view is further reinforced by studies connecting induction heads to in-context learning and repetition dynamics (Crosbie and Shutova, 2025; Wang et al., 2025b; Mahaut and Franzon, 2025). Taken together, these findings suggest that unbounded drift reflects a structured internal bias toward copying and repetition, which can be amplified into persistent looping once normal stopping behavior breaks down.

Behavioral Dynamics. At the behavioral level, repetition exhibits strong self-reinforcing dynamics once triggered. Pseudo-Repetition Penalization (Xu et al., 2022) shows that, after repetition begins, the probability of generating identical content rapidly increases, forming a closed loop. Complementarily, uncertainty-driven fallback modeling (Ivgi et al., 2024) interprets repetition as a fallback behavior under uncertainty, where models revert to low-entropy states by repeating prior text when unable to identify a plausible continuation.

Key Insights.

Unbounded drift stems from both model architecture and training methods. While decoding-time interventions offer immediate relief, training-level approaches may provide more stable, fundamental mitigation by addressing root causes rather than surface-level symptoms.

5 Mitigation Strategies for Resource Consumption Threats

This section reviews existing mitigation methods for threats to resource consumption. Importantly, these methods are not equally security-oriented: many aim to improve efficiency under benign conditions, whereas only a smaller subset explicitly defends against adversarial resource amplification. Detailed technical descriptions of mitigation strategies are provided in Appendix H.

5.1 Mitigating Overthinking

LLMs and RLLMs. For reasoning models, most existing mitigations are better understood as efficiency-oriented interventions rather than defenses explicitly designed against adversarial resource amplification. Existing overthinking is often studied as an inference-efficiency problem; current methods mainly aim to reduce unnecessary reasoning length under benign conditions, while only indirectly mitigating attack-induced overhead.

At the training level, these methods aim to foster awareness of the length of reasoning. Wu et al. (2025) introduce budget signals through reward shaping, while parameter-space tuning identifies controllable directions that compress reasoning chains within a single model (Ma et al., 2025).

At the decoding level, lightweight interventions suppress redundant reasoning without retraining, for example, by stopping unnecessary reflection once the model is sufficiently certain or by interrupting repetitive hidden-state patterns and injecting control vectors to regulate thinking depth (Huang et al., 2025b; Xie et al., 2025; Liu et al., 2025a; Lin et al., 2025; He and Tang, 2025).

At the system level, resource allocation is adjusted more explicitly. Difficulty-adaptive methods allocate inference budget according to query complexity (Wang et al., 2025e), while token-budget-aware prompting and monitoring-of-thought frameworks terminate unnecessary reasoning externally without modifying model parameters (Han et al., 2025; Zhu et al., 2025b).

Overall, these methods show that overthinking can be mitigated from multiple levels, but the current literature remains centered on efficiency optimization rather than attack-specific defense.

Multimodal Large Language Models (MLLMs). In multimodal settings, overthinking can be amplified by both unnecessary reasoning and large external contexts. Certainty-based routing (Lu et al., 2025) dynamically invokes CoT only when the model is uncertain, thereby avoiding prolonged reasoning on simpler inputs. Memory-augmented frameworks (Bhat et al., 2025; Ottem, 2025) further mitigate context bloat through structured memory modules and verification-centric filtering, significantly reducing context size without degrading retrieval quality.

Key Insights.

Existing mitigations have shown effectiveness in alleviating overthinking. However, current methods predominantly focus on efficiency-oriented mitigations rather than attack-aware defenses, and explicit mitigations for autonomous agent pipelines remain largely unexplored.

5.2 Mitigating Unbounded Drift

General LLMs. Unbounded drift in text generation often manifests as repetitive or uncontrolled decoding. As a result, mitigation strategies in general LLMs mainly focus on detecting or suppressing repetition to prevent outputs from degenerating into runaway generation loops.

At the model level, existing methods mitigate degeneration either through training-time objective design or decoding-time control. Unlikelihood training (Welleck et al., 2019) reduces degenerate text by penalizing repeated tokens, while RAP (Huang et al., 2025a) provides a structured way to tune repetition penalties during decoding while maintaining task performance.

At the system level, robustness is enforced through runtime detection and scheduling safeguards. For instance, RecurrentDetector (Yu et al., 2025) identifies repetitive activation patterns to halt infinite loops. Additionally, PD^{3}F (Zhang et al., 2025d) combines request scheduling with output EOS amplification. Queue-based architectures also help stabilize throughput under heavy workloads (Barek et al., 2025).

Overall, these methods show that uncontrolled repetitive generation in text models can be effectively mitigated through interventions across training, decoding, and system layers.

Reasoning Models. Dedicated defenses targeting non-terminating reasoning in RLLMs remain scarce. Existing techniques such as WSC (Xie et al., 2025) and self-affirmation suppression (Liu et al., 2025a) primarily focus on removing redundant reasoning during decoding, which can incidentally alleviate mild repetitive behaviors.

Agentic Systems. Agentic systems introduce additional vulnerabilities due to continuous perception pipelines. Background-attentive adversarial training (Wang et al., 2025c) improves robustness by incorporating perturbation-aware training objectives, effectively restoring inference speed and stability under resource abuse on edge devices.

Key Insights.

System-level frameworks effectively mitigate crash-level threats and DoS-style attacks in general LLMs. However, purpose-built crash defenses for MLLMs remain largely absent.

6 Open Challenges and Future Directions

6.1 From Efficiency Optimization to Security Guarantees

A useful distinction is between efficiency-oriented mitigation and security-oriented defense. The former treats excessive generation primarily as a cost problem under benign workloads, whereas the latter treats it as an adversarial resource-amplification threat that requires robustness guarantees. Much of the literature still frames excessive generation as a cost-latency trade-off, aiming to reduce average output length under benign conditions rather than defend against adversarial resource amplification (Jiang et al., 2024; Agrawal et al., 2024; Del Corro et al., 2023).

Looking forward, two research directions appear especially important. First, future research should move from heuristic efficiency control to security-oriented budget protection, treating resource usage as a constrained attack surface rather than merely an optimization target. Second, defenses should protect not only output-level resource usage but also process-level computation, including intermediate reasoning, tool interactions, and multi-turn execution trajectories. Together, these directions would help elevate resource control from an efficiency concern to a security objective for reliable and sustainable LLMs.

6.2 Understanding Threat Mechanisms

A major limitation of current research is the lack of a unified mechanistic understanding of resource amplification. Existing studies largely focus on specific attack instances, such as reasoning over-expansion or image-coded induction (Gao et al., 2025; Fu et al., 2025), while offering little general theory of how generation trajectories expand under attack manipulation. Future work should therefore integrate the shared dynamics and model-specific properties of different generative systems to develop transferable theories that support cross-model analysis and principled defense design.

6.3 Resource Abuse in LLM Ecosystems

LLMs expose an increasingly serious form of resource abuse, yet existing studies remain limited to a small number of attack surfaces (Zhang et al., 2025b; Zhou et al., 2026). Compared with model-level attacks, these threats are harder to characterize because resource consumption may accumulate across tool calls, intermediate services, memory updates, and multi-agent coordination, while the final output can still appear normal (Lee et al., 2026). Future research should therefore move beyond isolated case studies toward a systematic understanding of agent-level resource abuse, including component-wise resource accounting, cross-step dependency analysis, and defenses that can detect or interrupt malicious amplification throughout execution. This progress is increasingly important as agentic systems are deployed in real-world workflows, where resource abuse can directly undermine the reliability and sustainability of the services they support.

6.4 Toward Standardized Evaluation

A fundamental limitation of current research is the lack of a standardized evaluation framework for resource consumption threats. Existing studies span different model families, modalities, and deployment settings, but they often rely on heterogeneous metrics, which makes cross-study comparison difficult. To address this gap, Appendix F summarizes existing evaluation practices and presents a more unified evaluation framework. In particular, we argue that resource consumption threats should be assessed jointly from model-level behavior, hardware-level pressure, and application-level service impact, so that attacks and defenses can be compared under a common view. Without such a shared protocol, it remains difficult to consistently measure amplification severity, comprehensively evaluate defense coverage, or support downstream governance of resource consumption risks.

7 Conclusion

Resource consumption threats are emerging as an important safety concern for large language models. By inducing excessive and unnecessary generation, they not only reduce efficiency but also create broader risks for system reliability, service availability, and operational cost. This survey provided a unified view of the area through two representative regimes, *Overthinking* and *Unbounded Drift*, and reviewed existing work spanning attacks, mechanisms, defenses, and open challenges. Despite recent progress, the field remains fragmented in taxonomy, theory, evaluation, and mitigation. We hope this survey provides a clearer foundation for future research on understanding and mitigating threats to resource consumption and helps advance safer, more sustainable LLMs.

Limitations

This survey has several limitations. First, our discussion centers on *resource consumption threats in LLMs*, rather than the broader landscape of efficiency, robustness, or availability problems. In particular, we primarily focus on malicious or adversarial resource abuse that induces excessive generation, and therefore do not aim to comprehensively cover benign efficiency optimization, general systems engineering for acceleration, or all forms of denial-of-service behavior outside the generative process itself. As a result, some closely related work on inference acceleration, scheduling, or hardware optimization is discussed only when it directly informs the security perspective of resource consumption.

Second, the taxonomy proposed in this survey is intended as a unifying conceptual abstraction rather than a complete partition of all possible failure modes. While this distinction helps organize existing studies by the evolution pattern of generation, some attacks may exhibit mixed characteristics, shift between the two regimes across settings, or involve multiple system components simultaneously. This issue is especially relevant in agentic systems, where resource amplification may accumulate across tool calls, memory updates, retrieval steps, and multi-agent coordination, making strict categorization more difficult.

Third, the literature in this area remains highly uneven across model families and attack settings. Existing studies are concentrated on text-based LLMs and a relatively small set of open-source models, while black-box commercial systems remain harder to study systematically. Coverage is also imbalanced across modalities and deployment scenarios: current multimodal work still focuses heavily on image-centered settings, whereas the risks in audio and video modalities remain far less explored; similarly, many downstream agent applications have not yet been systematically examined from the perspective of resource consumption.

Finally, because this is an emerging field, the available empirical evidence remains fragmented, and evaluation practices are not yet standardized. Many papers report attack success using different metrics, threat models, and deployment assumptions, making direct comparison difficult and potentially limiting the stability of any unified conclusions. For this reason, our synthesis should be understood as a structured overview of the current research landscape rather than a definitive benchmark of attack prevalence, mechanism universality, or defense effectiveness.

Ethical Considerations

This survey reviews resource consumption attacks on LLMs. Although summarizing attack mechanisms may increase understanding of such threats, our goal is strictly defensive: to clarify the threat landscape, support standardized evaluation, and encourage effective mitigation. We therefore focus on conceptual analysis and system-level implications rather than actionable attack instructions.

We further emphasize that resource consumption abuse can have real-world consequences beyond efficiency degradation. In shared LLM infrastructure, excessive generation may reduce service availability, increase operational costs, and harm end users through higher latency, unstable access, and higher usage costs. These risks can become even more severe in agentic systems, where recursive execution and covert workload amplification may disrupt workflows and cause hidden financial loss. By synthesizing this emerging area, we aim to support safer, more reliable, and more sustainable LLMs.

References

Amey Agrawal, Nitin Kedia, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav Gulavani, Alexey Tumanov, and Ramachandran Ramjee. 2024. Taming {Throughput-Latency} tradeoff in {LLM} inference with {Sarathi-Serve}. In *18th USENIX symposium on operating systems design and implementation (OSDI 24)*, pages 117–134.

Soogand Alavi, Salar Nozari, and Andrea Luangrath. 2025. Cost transparency of enterprise ai adoption. *arXiv preprint arXiv:2511.11761*.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, and 1 others. 2022. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736.

Sotiris Anagnostidis, Dario Pavllo, Luca Biggio, Lorenzo Noci, Aurelien Lucchi, and Thomas Hofmann. 2023. Dynamic context pruning for efficient and interpretable autoregressive transformers. *Advances in Neural Information Processing Systems*, 36:65202–65223.

Anthropic. 2025. Claude code.

Amit Baras, Alon Zolfi, Yuval Elovici, and Asaf Shabtai. 2025. Quantattack: Exploiting quantization techniques to attack vision transformers. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 6730–6740. IEEE.

MD Abdul Barek, Md Bajlur Rashid, Md Mostafizur Rahman, ABM Kamrul Islamc Riad, Guillermo Francia, Hossain Shahriar, and Sheikh Iqbal Ahamed. 2025. Vulnerability to stability: Scalable large language model in queue-based web service. In *2025 IEEE 49th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 995–1000. IEEE.

Dirk Bergemann, Alessandro Bonatti, and Alex Smolin. 2025. The economics of large language models: Token allocation, fine-tuning, and optimal pricing. In *Proceedings of the 26th ACM Conference on Economics and Computation*, pages 786–786.

Sachin Bhat, Erik Cambria, and Haibo Peng. 2025. Nmret: A memory-augmented retrieval framework for large language models. Technical report, SenticNet / Nanyang Technological University.

Chen and 1 others. 2021. Translowdown: Efficiency attacks on neural machine translation systems.

Erh-Chung Chen, Pin-Yu Chen, I-Hsin Chung, and Che-Rung Lee. 2024. Overload: Latency attacks on object detection for edge devices. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 24716–24725.

Guangyao Chen, Siwei Dong, Yu Shu, Ge Zhang, Jaward Sesay, Börje F Karlsson, Jie Fu, and Yemin Shi. 2023a. Autoagents: A framework for automatic agent generation. *arXiv preprint arXiv:2309.17288*.

Simin Chen, Hanlin Chen, Mirazul Haque, Cong Liu, and Wei Yang. 2023b. The dark side of dynamic routing neural networks: Towards efficiency backdoor injection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24585–24594.

Simin Chen, Cong Liu, Mirazul Haque, Zihe Song, and Wei Yang. 2022a. Nmtsloth: understanding and testing efficiency degradation of neural machine translation systems. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 1148–1160.

Simin Chen, Zihe Song, Mirazul Haque, Cong Liu, and Wei Yang. 2022b. Nicgslowdown: Evaluating the efficiency robustness of neural image caption generation models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15365–15374.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. 2025. Do NOT think that much for 2+3=? on the overthinking of long reasoning models. In *Forty-second International Conference on Machine Learning*.

Antonio Emanuele Cinà, Ambra Demontis, Battista Biggio, Fabio Roli, and Marcello Pelillo. 2025. Energy-latency attacks via sponge poisoning. *Information Sciences*, 702:121905.

Antonio Emanuele Cinà, Sebastiano Vascon, Ambra Demontis, Battista Biggio, Fabio Roli, and Marcello Pelillo. 2021. The hammer and the nut: Is bilevel optimization really needed to poison linear classifiers? In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Antonia Creswell and Murray Shanahan. 2022. Faithful reasoning using large language models. *arXiv preprint arXiv:2208.14271*.

Joy Crosbie and Ekaterina Shutova. 2025. Induction heads as an essential mechanism for pattern matching in in-context learning. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 5034–5096.

Luciano Del Corro, Allie Del Giorno, Sahaj Agarwal, Bin Yu, Ahmed Awadallah, and Subhabrata Mukherjee. 2023. Skipdecode: Autoregressive skip decoding with batching and caching for efficient llm inference. *arXiv preprint arXiv:2307.02628*.

Nhi Hoai Doan, Tatsuya Hiraoka, and Kentaro Inui. 2025. Understanding and controlling repetition neurons and induction heads in in-context learning. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pages 2854–2876.

Ben Dong, Hui Feng, and Qian Wang. 2026. Clawdrain: Exploiting tool-calling chains for stealthy token exhaustion in openclaw agents. *Preprint*, arXiv:2603.00902.

Jianshuo Dong, Ziyuan Zhang, Qingjie Zhang, Tianwei Zhang, Hao Wang, Hewu Li, Qi Li, Chao Zhang, Ke Xu, and Han Qiu. 2024. An engorgio prompt makes large language model babble on. *arXiv preprint arXiv:2412.19394*.

Erin Feiglin, Nir Hutnik, and Raz Lapid. 2026. Benchoverflow: Measuring overflow in large language models via plain-text prompts. *arXiv preprint arXiv:2601.08490*.

Xiaoning Feng, Xiaohong Han, Simin Chen, and Wei Yang. 2024. Llmeffichecker: Understanding and testing efficiency degradation of large language models. *ACM Transactions on Software Engineering and Methodology*, 33(7):1–38.

Jared Fernandez, Clara Na, Vashisth Tiwari, Yonatan Bisk, Sasha Luccioni, and Emma Strubell. 2025. Energy considerations of large language model inference and efficiency optimizations. In *Proceedings*

of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 32556–32569.

Jiyuan Fu, Kaixun Jiang, Lingyi Hong, Jinglun Li, Haijing Guo, Dingkang Yang, Zhaoyu Chen, and Wenqiang Zhang. 2025. Lingoloop attack: Trapping mllms via linguistic context and state entrapment into endless loops. *arXiv preprint arXiv:2506.14493*.

Zihao Fu, Wai Lam, Anthony Man-Cho So, and Bei Shi. 2021. A theoretical analysis of the repetition problem in text generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 12848–12856.

Pascale Fung, Yoram Bachrach, Asli Celikyilmaz, Kamalika Chaudhuri, Delong Chen, Willy Chung, Emmanuel Dupoux, Hongyu Gong, Hervé Jégou, Alessandro Lazaric, and 1 others. 2025. Embodied ai agents: Modeling the world. *arXiv preprint arXiv:2506.22355*.

Haoran Gao, Yuanhe Zhang, Zhenhong Zhou, Lei Jiang, Fanyu Meng, Yujia Xiao, Li Sun, Kun Wang, Yang Liu, and Junlan Feng. 2025. Resource consumption red-teaming for large vision-language models. *arXiv preprint arXiv:2507.18053*.

Kuofeng Gao, Yang Bai, Jindong Gu, Shu-Tao Xia, Philip Torr, Zhifeng Li, and Wei Liu. 2024a. Inducing high energy-latency of large vision-language models with verbose images. *arXiv preprint arXiv:2401.11170*.

Kuofeng Gao, Jindong Gu, Yang Bai, Shu-Tao Xia, Philip Torr, Wei Liu, and Zhifeng Li. 2024b. Energy-latency manipulation of multi-modal large language models via verbose samples. *arXiv preprint arXiv:2404.16557*.

Kuofeng Gao, Tianyu Pang, Chao Du, Yong Yang, Shu-Tao Xia, and Min Lin. 2024c. Denial-of-service poisoning attacks against large language models. *arXiv preprint arXiv:2410.10760*.

Jonas Geiping, Alex Stein, Manli Shu, Khalid Saifullah, Yuxin Wen, and Tom Goldstein. 2024. Coercing llms to do and reveal (almost) anything, 2024. *URL https://arxiv. org/abs/2402.14020*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, and 1 others. 2025a. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Yi Guo, Kanchana Thilakarathna, Nirhoshan Sivaroopan, Jo Plested, Tim Lynar, Jack Yang, Wangli Yang, and 1 others. 2025b. Prompt-induced over-generation as denial-of-service: A black-box attack-side benchmark. *arXiv preprint arXiv:2512.23779*.

Ghaith Hammouri, Kemal Derya, and Berk Sunar. 2025. Non-halting queries: Exploiting fixed points in llms. In *2025 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pages 1–22. IEEE.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. 2025. Token-budget-aware llm reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 24842–24855.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. 2023. Reasoning with language model is planning with world model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8154–8173.

Syed Mhamudul Hasan, Hussein Zangoti, Iraklis Anagnostopoulos, and Abdur R Shahid. 2025a. Sponge attacks on sensing ai: Energy-latency vulnerabilities and defense via model pruning. *arXiv preprint arXiv:2505.06454*.

Syed Mhamudul Hasan, Hussein Zangoti, Iraklis Anagnostopoulos, and Abdur R. Shahid. 2025b. Sponge attacks on sensing ai: Energy-latency vulnerabilities and defense via model pruning. *Preprint*, arXiv:2505.06454.

Jujie He, Jiacai Liu, Chris Yuhao Liu, Rui Yan, Chaojie Wang, Peng Cheng, Xiaoyu Zhang, Fuxiang Zhang, Jiacheng Xu, Wei Shen, and 1 others. 2025. Skywork open reasoner 1 technical report. *arXiv preprint arXiv:2505.22312*.

Yixin He and Lumingyuan Tang. 2025. Learning to ponder: Adaptive reasoning in latent space. *arXiv preprint arXiv:2509.24238*.

Tatsuya Hiraoka and Kentaro Inui. 2025. Repetition neurons: How do language models produce repetitions? In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 483–495.

Sanghyun Hong, Yiğitcan Kaya, Ionuţ-Vlad Modoranu, and Tudor Dumitraş. 2020. A panda? no, it's a sloth: Slowdown attacks on adaptive multi-exit neural network inference. *arXiv preprint arXiv:2010.02432*.

Donghao Huang, Thanh-Son Nguyen, Fiona Liausvia, and Zhaoxia Wang. 2025a. Rap: A metric for balancing repetition and performance in open-source large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1479–1496.

Jiameng Huang, Baijiong Lin, Guhao Feng, Jierun Chen, Di He, and Lu Hou. 2025b. Efficient reasoning for large reasoning language models via certainty-guided reflection suppression. *arXiv preprint arXiv:2508.05337*.

Junquan Huang, Haotian Wu, Yubo Gao, Yibo Yan, Junyan Zhang, Yonghua Hei, Song Dai, Jie Zhang, Puay Siew Tan, and Xuming Hu. 2025c. Effireasonbench: A unified benchmark for evaluating and advancing efficient reasoning in large language models. *arXiv preprint arXiv:2511.10201*.

Ruixuan Huang, Qingyue Wang, Hantao Huang, Yudong Gao, Dong Chen, Shuai Wang, and Wei Wang. 2025d. Repetitioncurse: Measuring and understanding router imbalance in mixture-of-experts llms under dos stress. *Preprint*, arXiv:2512.23995.

Muhammed Emrullah Ildiz, Yixiao Huang, Yingcong Li, Ankit Singh Rawat, and Samet Oymak. 2024. From self-attention to markov models: Unveiling the dynamics of generative transformers. In *International Conference on Machine Learning*, pages 20955–20982. PMLR.

Maor Ivgi, Ori Yoran, Jonathan Berant, and Mor Geva. 2024. From loops to oops: Fallback behaviors of language models under uncertainty. *arXiv preprint arXiv:2407.06071*.

Fengqing Jiang, Zhangchen Xu, Yuetai Li, Luyao Niu, Zhen Xiang, Bo Li, Bill Yuchen Lin, and Radha Poovendran. 2025. Safechain: Safety of language models with long chain-of-thought reasoning capabilities. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 23303–23320.

Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2024. Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1658–1677.

Abhinav Kumar, Jaechul Roh, Ali Naseh, Marzena Karpinska, Mohit Iyyer, Amir Houmansadr, and Eugene Bagdasarian. 2025. Overthink: Slowdown attacks on reasoning llms. *Preprint*, arXiv:2502.02542.

Yohan Lee, Jisoo Jang, Seoyeon Choi, Sangyeop Kim, and Seungtaek Choi. 2026. Overthinking loops in agents: A structural risk via mcp tools. *arXiv preprint arXiv:2602.14798*.

Huayang Li, Tian Lan, Zihao Fu, Deng Cai, Lemao Liu, Nigel Collier, Taro Watanabe, and Yixuan Su. 2023. Repetition in repetition out: Towards understanding neural text degeneration from the data perspective. *Advances in Neural Information Processing Systems*, 36:72888–72903.

Xingyu Li, Xiaolei Liu, Cheng Liu, Yixiao Xu, Kangyi Ding, Bangzhou Xin, and Jia-Li Yin. 2025a. Loopllm: Transferable energy-latency attacks in llms via repetitive generation. *arXiv preprint arXiv:2511.07876*.

Yunzhe Li, Jianan Wang, Hongzi Zhu, James Lin, Shan Chang, and Minyi Guo. 2025b. Thinktrap: Denial-of-service attacks against black-box llm services via infinite thinking. *arXiv preprint arXiv:2512.07086*.

Xiang Lin, Simeng Han, and Shafiq Joty. 2021. Straight to the gradient: Learning to use novel tokens for neural text generation. In *International conference on machine learning*, pages 6642–6653. PMLR.

Zhengkai Lin, Zhihang Fu, Ze Chen, Chao Chen, Liang Xie, Wenxiao Wang, Deng Cai, Zheng Wang, and Jieping Ye. 2025. Controlling thinking speed in reasoning models. *arXiv preprint arXiv:2507.03704*.

Jona te Lintelo, Stefanos Koffas, and Stjepan Picek. 2024. The skipsponge attack: Sponge weight poisoning of deep neural networks. *arXiv preprint arXiv:2402.06357*.

Han Liu, Yuhao Wu, Zhiyuan Yu, Yevgeniy Vorobeychik, and Ning Zhang. 2023. Slowlidar: Increasing the latency of lidar-based detection using adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5146–5155.

Kaiyuan Liu, Chen Shen, Zhanwei Zhang, Junjie Liu, Xiaosong Yuan, and 1 others. 2025a. Efficient reasoning through suppression of self-affirmation reflections in large reasoning models. *arXiv preprint arXiv:2506.12353*.

Shuaitong Liu, Renjue Li, Lijia Yu, Lijun Zhang, Zhiming Liu, and Gaojie Jin. 2025b. Badthink: Triggered overthinking attacks on chain-of-thought reasoning in large language models. *Preprint*, arXiv:2511.10714.

Nitin Lodha. 2025. Tokenops: A compiler-style architecture for token optimization in llm api workflows. Technical report, Chitrangana.com.

Jiahao Lu, Yifan Zhang, Qiuhong Shen, Xinchao Wang, and Shuicheng Yan. 2024. Poison-splat: Computation cost attack on 3d gaussian splatting. *arXiv preprint arXiv:2410.08190*.

Jinghui Lu, Haiyang Yu, Siliang Xu, Shiwei Ran, Guozhi Tang, Siqi Wang, Bin Shan, Teng Fu, Hao Feng, Jingqun Tang, and 1 others. 2025. Prolonged reasoning is not all you need: Certainty-based adaptive routing for efficient llm/mllm reasoning. *arXiv preprint arXiv:2505.15154*.

Chen Ma, Ningfei Wang, Qi Alfred Chen, and Chao Shen. 2024. Slowtrack: Increasing the latency of camera-based perception in autonomous driving using adversarial examples. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(5):4062–4070.

Xinyin Ma, Guangnian Wan, Runpeng Yu, Gongfan Fang, and Xinchao Wang. 2025. Cot-valve: Length-compressible chain-of-thought tuning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6025–6035.

Matéo Mahaut and Francesca Franzon. 2025. Repetitions are not all alike: distinct mechanisms sustain repetition in language models. *arXiv preprint arXiv:2504.01100*.

Eran Malach. 2024. Auto-regressive next-token predictors are universal learners. In *Proceedings of the 41st International Conference on Machine Learning*, pages 34417–34431.

Hanene FZ Brachemi Meftah, Wassim Hamidouche, Sid Ahmed Fezza, Olivier Déforges, and Kassem Kallas. 2025. Energy backdoor attack to deep neural networks. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE.

Xupeng Miao, Chunan Shi, Jiangfei Duan, Xiaoli Xi, Dahua Lin, Bin Cui, and Zhihao Jia. 2024. Spotserve: Serving generative large language models on preemptible instances. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, pages 1112–1127.

Andreas Müller and Erwin Quiring. 2024. The impact of uniform inputs on activation sparsity and energy-latency attacks in computer vision. In *2024 IEEE Security and Privacy Workshops (SPW)*, pages 104–111. IEEE.

Raymond Muller, Ruoyu Song, Chenyi Wang, Yuxia Zhan, Jean-Phillipe Monteuuis, Yanmao Man, Ming Li, Ryan Gerdes, Jonathan Petit, and Z. Berkay Celik. 2025. Investigating physical latency attacks against camera-based perception. In *2025 IEEE Symposium on Security and Privacy (SP)*, pages 4588–4605.

Sania Nayab, Giulio Rossolini, Marco Simoni, Andrea Saracino, Giorgio Buttazzo, Nicolamaria Manes, and Fabrizio Giacomelli. 2024. Concise thoughts: Impact of output length on llm reasoning and cost. *arXiv preprint arXiv:2407.19825*.

Andreas Ottem. 2025. Meve: A modular system for memory verification and effective context control in language models. In *CS & IT Conference Proceedings*, volume 15. CS & IT Conference Proceedings.

Vijay Janapa Reddi. 2025. Generative ai at the edge: challenges and opportunities: the next phase in ai deployment. *Queue*, 23(2):79–137.

Kerem Sahin, Sheridan Feucht, Adam Belfki, Jannik Brinkmann, Aaron Mueller, David Bau, and Chris Wendler. 2025. In-context learning without copying. *arXiv preprint arXiv:2511.05743*.

Siddharth Samsi, Dan Zhao, Joseph McDonald, Baolin Li, Adam Michaleas, Michael Jones, William Bergeron, Jeremy Kepner, Devesh Tiwari, and Vijay Gadepally. 2023. From words to watts: Benchmarking the energy costs of large language model inference. In *2023 IEEE high performance extreme computing conference (HPEC)*, pages 1–9. IEEE.

Coen Schoof, Stefanos Koffas, Mauro Conti, and Stjepan Picek. 2024. Beyond phantomsponges: Enhancing sponge attack on object detection models. In *Proceedings of the 2024 ACM Workshop on Wireless Security and Machine Learning*, pages 14–19.

Minjae Seo, Myoungsung You, Junhee Lee, Jaehan Kim, Hwanjo Heo, Jintae Oh, and Jinwoo Kim. 2025. Eo-vlm: Vlm-guided energy overload attacks on vision models. *arXiv preprint arXiv:2504.08205*.

Avishag Shapira, Alon Zolfi, Luca Demetrio, Battista Biggio, and Asaf Shabtai. 2023. Phantom sponges: Exploiting non-maximum suppression to attack deep object detectors. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 4571–4580.

Natalie Shapira, Chris Wendler, Avery Yen, Gabriele Sarti, Koyena Pal, Olivia Floody, Adam Belfki, Alex Loftus, Aditya Ratan Jannali, Nikhil Prakash, Jasmine Cui, Giordano Rogers, Jannik Brinkmann, Can Rager, Amir Zur, Michael Ripa, Aruna Sankaranarayanan, David Atkinson, Rohit Gandikota, and 19 others. 2026. Agents of chaos. *Preprint*, arXiv:2602.20021.

Ilia Shumailov, Yiren Zhao, Daniel Bates, Nicolas Papernot, Robert Mullins, and Ross Anderson. 2021. Sponge examples: Energy-latency attacks on neural networks. In *2021 IEEE European symposium on security and privacy (EuroS&P)*, pages 212–231. IEEE.

Wai Man Si, Michael Backes, and Yang Zhang. 2023. Mondrian: Prompt abstraction attack against large language models for cheaper api pricing. *arXiv preprint arXiv:2308.03558*.

Wai Man Si, Mingjie Li, Michael Backes, and Yang Zhang. 2025. Excessive reasoning attack on reasoning llms. *Preprint*, arXiv:2506.14374.

Peter Steinberger. 2026. Openclaw: Personal ai assistant.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, and 1 others. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*.

Guoheng Sun, Ziyao Wang, Bowei Tian, Meng Liu, Zheyu Shen, Shwai He, Yexiao He, Wanghao Ye, Yiting Wang, and Ang Li. 2025. Coin: Counting the invisible reasoning tokens in commercial opaque llm apis. *arXiv preprint arXiv:2505.13778*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Chenyi Wang, Ruoyu Song, Raymond Muller, Jean-Philippe Monteuuis, Z. Berkay Celik, Jonathan Petit, Ryan Gerdes, and Ming Li. 2025a. Cp-freezer: Latency attacks against vehicular cooperative perception. *Preprint*, arXiv:2508.01062.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023a. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, and 1 others. 2024a. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345.

Shuxun Wang, Qingyu Yin, Chak Tou Leong, Qiang Zhang, and Linyi Yang. 2025b. Induction head toxicity mechanistically explains repetition curse in large language models. *arXiv preprint arXiv:2505.13514*.

Siqi Wang, Hailong Yang, Xuezhu Wang, Tongxuan Liu, Pengbo Wang, Xuning Liang, Kejie Ma, Tianyu Feng, Xin You, Yongjun Bao, and 1 others. 2024b. Minions: Accelerating large language model inference with aggregated speculative execution. *arXiv preprint arXiv:2402.15678*.

Tianyi Wang, Huawei Fan, Yuanchao Shu, Peng Cheng, and Cong Wang. 2026a. Rethinking latency denial-of-service: Attacking the llm serving framework, not the model. *arXiv preprint arXiv:2602.07878*.

Tianyi Wang, Zichen Wang, Cong Wang, Yuanchao Shu, Ruilong Deng, Peng Cheng, and Jiming Chen. 2025c. Can't slow me down: Learning robust and hardware-adaptive object detectors against latency attacks for edge devices. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 19230–19240.

Xiasi Wang, Tianliang Yao, Simin Chen, Runqi Wang, Lei Ye, Kuofeng Gao, Yi Huang, and Yuan Yao. 2025d. Vlminferslow: Evaluating the efficiency robustness of large vision-language models as a service. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16035–16050.

Xinglin Wang, Shaoxiong Feng, Yiwei Li, Peiwen Yuan, Yueqi Zhang, Chuyi Tan, Boyuan Pan, Yao Hu, and Kan Li. 2025e. Make every penny count: Difficulty-adaptive self-consistency for cost-efficient reasoning. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 6904–6917.

Yanlin Wang, Jiadong Wu, Tianyue Jiang, Mingwei Liu, Jiachi Chen, Chong Wang, Ensheng Shi, Xilin Liu, Yuchi Ma, and Zibin Zheng. 2025f. Draincode: Stealthy energy consumption attacks on retrieval-augmented code generation via context poisoning. In *2025 40th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 778–790. IEEE.

Zijian Wang, Shuo Huang, Yujin Huang, and Helei Cui. 2023b. Energy-latency attacks to on-device neural networks via sponge poisoning. In *Proceedings of the 2023 Secure and Trustworthy Deep Learning Systems Workshop*, pages 1–11.

Ziwei Wang, Yuanhe Zhang, Jing Chen, Zhenhong Zhou, Ruichao Liang, Ruiying Du, Ju Jia, Cong Wu, and Yang Liu. 2026b. Recur: Resource exhaustion attack via recursive-entropy guided counterfactual utilization and reflection. *arXiv preprint arXiv:2602.08214*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. 2019. Neural text generation with unlikelihood training. *arXiv preprint arXiv:1908.04319*.

Xingyu Wu, Yuchen Yan, Shangke Lyu, Linjuan Wu, Yiwen Qiu, Yongliang Shen, Weiming Lu, Jian Shao, Jun Xiao, and Yueting Zhuang. 2025. Lapo: Internalizing reasoning efficiency via length-adaptive policy optimization. *arXiv preprint arXiv:2507.15758*.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, and 1 others. 2025. The rise and potential of large language model based agents: A survey. *Science China Information Sciences*, 68(2):121101.

Wenya Xie, Shaochen Zhong, Hoang Anh Duy Le, Zhaozhuo Xu, Jianwen Xie, and Zirui Liu. 2025. Word salad chopper: Reasoning models waste a ton of decoding budget on useless repetitions, self-knowingly. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 33576–33586.

Jin Xu, Xiaojiang Liu, Jianhao Yan, Deng Cai, Huayang Li, and Jian Li. 2022. Learning to break the loop: Analyzing and mitigating repetitions for neural text generation. *Advances in Neural Information Processing Systems*, 35:3082–3095.

Xiaobei Yan, Yiming Li, Hao Wang, Han Qiu, and Tianwei Zhang. 2025. Bithydra: Towards bit-flip inference cost attack against large language models. *arXiv preprint arXiv:2505.16670*.

Junchi Yao, Shu Yang, Jianhua Xu, Lijie Hu, Mengdi Li, and Di Wang. 2025. Understanding the repeat curse in large language models from a feature perspective. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 7787–7815, Vienna, Austria. Association for Computational Linguistics.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Biao Yi, Zekun Fei, Jianing Geng, Tong Li, Lihai Nie, Zheli Liu, and Yiming Li. 2025. Badreasoner: Planting tunable overthinking backdoors into

large reasoning models for fun or profit. *Preprint*, arXiv:2507.18305.

Itay Yona, Ilia Shumailov, Jamie Hayes, Federico Barbero, and Yossi Gandelsman. 2025a. Interpreting the repeated token phenomenon in large language models. *arXiv preprint arXiv:2503.08908*.

Itay Yona, Ilia Shumailov, Jamie Hayes, and Yossi Gandelsman. 2025b. Interpreting the repeated token phenomenon in large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 72535–72555. PMLR.

Junzhe Yu, Yi Liu, Huijia Sun, Ling Shi, and Yuqi Chen. 2025. Breaking the loop: Detecting and mitigating denial-of-service vulnerabilities in large language models. *arXiv preprint arXiv:2503.00416*.

Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. 2024. Injecagent: Benchmarking indirect prompt injections in tool-integrated large language model agents. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 10471–10506.

Rui Zhang, Zihan Wang, Tianli Yang, Hongwei Li, Wenbo Jiang, Qingchuan Zhao, Yang Liu, and Guowen Xu. 2025a. Hidden tail: Adversarial image causing stealthy resource consumption in vision-language models. *arXiv preprint arXiv:2508.18805*.

Yuanhe Zhang, Weiliu Wang, Zhenhong Zhou, Kun Wang, Jie Zhang, Li Sun, Yang Liu, and Sen Su. 2025b. Leechhijack: Covert computational resource exploitation in intelligent agent systems. *Preprint*, arXiv:2512.02321.

Yuanhe Zhang, Weiliu Wang, Zhenhong Zhou, Kun Wang, Jie Zhang, Li Sun, Yang Liu, and Sen Su. 2025c. Leechhijack: Covert computational resource exploitation in intelligent agent systems. *arXiv preprint arXiv:2512.02321*.

Yuanhe Zhang, Xinyue Wang, Haoran Gao, Zhenhong Zhou, Fanyu Meng, Yuyao Zhang, and Sen Su. 2025d. pd3f: A pluggable and dynamic dos-defense framework against resource consumption attacks targeting large language models. *arXiv preprint arXiv:2505.18680*.

Yuanhe Zhang, Zhenhong Zhou, Wei Zhang, Xinyue Wang, Xiaojun Jia, Yang Liu, and Sen Su. 2025e. Crabs: Consuming resource via auto-generation for llm-dos attack under black-box settings. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 11128–11150.

Shuli Zhao, Qinsheng Hou, Zihan Zhan, Yanhao Wang, Yuchong Xie, Yu Guo, Libo Chen, Shenghong Li, and Zhi Xue. 2025. Mind your server: A systematic study of parasitic toolchain attacks on the mcp ecosystem. *arXiv preprint arXiv:2509.06572*.

Yue Zheng, Yuhao Chen, Bin Qian, Xiufang Shi, Yuanchao Shu, and Jiming Chen. 2025. A review on edge large language models: Design, execution, and applications. *ACM Comput. Surv.*, 57(8).

Kaiyu Zhou, Yongsen Zheng, Yicheng He, Meng Xue, Xueluan Gong, Yuji Wang, and Kwok-Yan Lam. 2026. Beyond max tokens: Stealthy resource amplification via tool calling chains in llm agents. *arXiv preprint arXiv:2601.10955*.

Zhenhong Zhou, Zherui Li, Jie Zhang, Yuanhe Zhang, Kun Wang, Yang Liu, and Qing Guo. 2025. Corba: Contagious recursive blocking attacks on multi-agent systems based on large language models. *Preprint*, arXiv:2502.14529.

Zixuan Zhou, Xuefei Ning, Ke Hong, Tianyu Fu, Jiaming Xu, Shiyao Li, Yuming Lou, Luning Wang, Zhihang Yuan, Xiuhong Li, and 1 others. 2024. A survey on efficient inference for large language models. *arXiv preprint arXiv:2404.14294*.

Zhenhao Zhu, Yue Liu, Zhiwei Xu, Yingwei Ma, Hongcheng Gao, Nuo Chen, Yanpei Guo, Wenjie Qu, Huiying Xu, Zifeng Kang, Xinzhong Zhu, and Jiaheng Zhang. 2025a. Extendattack: Attacking servers of lrms via extending reasoning. *Preprint*, arXiv:2506.13737.

Zihao Zhu, Hongbao Zhang, Ruotong Wang, Ke Xu, Siwei Lyu, and Baoyuan Wu. 2025b. To think or not to think: Exploring the unthinking vulnerability in large reasoning models. *arXiv preprint arXiv:2502.12202*.

A Scope and Inclusion Criteria

This survey focuses on threats to resource consumption in LLMs. Our primary focus is on adversarial or malicious resource abuse that increases computational cost through extended, excessive, or uncontrolled generation. Accordingly, we treat attacks on LLMs, reasoning models, multimodal generative systems, and agentic pipelines as the core literature when they induce abnormal output length, non-convergent decoding, recursive execution expansion, or related forms of resource amplification.

We also discuss a limited set of closely related works from vision and systems security, including selected latency and sponge-style attacks, but only in a supporting role. These works are included when they provide historical precursors, analogous threat patterns, or mechanistic insights that help explain later resource abuse in generative systems. They should not be interpreted as equally central instances of the problem studied in this survey.

By contrast, we do not aim to comprehensively review benign efficiency optimization, generic inference acceleration, hardware scheduling, or all forms of denial-of-service and availability attacks

outside the generative process itself. Such work is referenced only when it directly informs the security perspective on resource consumption threats or helps contextualize the survey's boundary.

B Preliminary

B.1 Autoregressive Generation

Most modern LLMs produce outputs through an autoregressive decoding process (Vaswani et al., 2017; Malach, 2024). Given an input prompt x, the model generates an output sequence $y = (y_1, y_2, \ldots, y_T)$ token by token according to the conditional distribution:

$$p(y_t \mid x, y_{<t}), \tag{1}$$

where $y_{<t}$ denotes the previously generated tokens and T is the final generation length. At each decoding step, the model computes a probability distribution over the vocabulary conditioned on the prompt and previously generated tokens, and selects the next token via sampling or deterministic decoding.

The decoding process continues iteratively until a termination condition is satisfied. Common termination criteria include emitting a special end-of-sequence token, reaching a predefined maximum length, or satisfying task-specific stopping rules. Consequently, the overall generation process can be viewed as a sequence of decoding steps, each extending the partial output and updating the model's internal state.

Because each token requires an additional decoding step, the generation length T directly determines the number of forward passes executed during inference. As a result, the computational cost of a generation process grows with the length of the generated sequence. This characteristic makes generation length a central factor in determining the resource usage of modern LLMs.

B.2 Computational Cost of Generation

The computational cost of autoregressive generation arises from repeated forward passes through large neural networks. For a model with parameters θ, generating a sequence of length T requires evaluating the model T times to compute the conditional probabilities $p(y_t \mid x, y_{<t}; \theta)$.

In addition to repeated model evaluation, attention-based architectures introduce further computational overhead. Self-attention mechanisms require each token to attend to all previously generated tokens, leading to increased memory usage and computation as the sequence length increases. Consequently, the total computational cost of generation can be approximated as a function of the output length:

$$C(T) = \sum_{t=1}^{T} c_t, \tag{2}$$

where c_t denotes the cost of the t-th decoding step. In practice, c_t may grow as the context window expands, leading to increasing latency and memory consumption for longer sequences.

These computational characteristics directly translate into operational costs in real-world deployments. Longer generation sequences lead to higher GPU utilization, increased inference latency, and greater energy consumption. For this reason, many commercial LLM services use token-based pricing models, charging users based on the number of generated tokens. Under such pricing schemes, longer outputs correspond directly to higher economic costs.

Therefore, generation length becomes a critical resource variable in LLMs. Any behavior that unnecessarily increases the number of generated tokens may significantly amplify computational cost and degrade system throughput.

B.3 Generation Trajectories

The decoding process can be viewed as a trajectory that describes how the model progresses from an input prompt to a final output. Formally, a generation process can be represented as a trajectory:

$$\tau = (x, y_1, y_2, \ldots, y_T), \tag{3}$$

which evolves over successive decoding steps.

Under normal conditions, the trajectory converges once the model produces a complete response that satisfies the task objective. However, the dynamics of this trajectory may be altered by adversarial prompts, system-level manipulations, or unexpected interaction patterns. In such cases, the generation trajectory may extend significantly beyond the length required for task completion.

Because computational cost is tightly coupled with the trajectory length, deviations in trajectory dynamics can directly amplify resource consumption. Understanding how generation trajectories evolve is therefore essential for analyzing resource consumption behaviors in LLMs.

Model	Context Window
GPT-5.4 / GPT-5.4-pro	1M
Claude Opus 4.6	200K (1M beta)
Claude Sonnet 4.6	200K (1M beta)
Gemini 2.5 Pro	1M
Gemini 2.5 Flash	1M
Mistral Large 2	128K
DeepSeek-V3 / R1	128K

Table 1: Historical Growth of Context Windows in Mainstream and Frontier Models

C The Rapid Growth of Context and Cost in LLM

Recent LLMs have expanded rapidly in both capability and serving scale, with context windows and deployment budgets increasing in parallel. As shown in Table 2, current mainstream APIs already exhibit substantial variation in usage costs, especially for high-end, reasoning-oriented, and frontier models, whose output-token pricing remains significantly higher than that of lightweight alternatives. At the same time, Table 1 shows that the context windows of representative commercial models have reached 128K, 200K, and even 1M tokens, making long-context generation an increasingly common operating regime rather than a special-case setting. This trend is further reflected in Table 3, where both mainstream and frontier models exhibit rapid year-over-year growth in supported context length.

Taken together, these trends indicate that modern LLMs are no longer optimized solely for short-response interactions but are increasingly designed to support long-horizon reasoning, retrieval, and multi-step execution. As context windows and output budgets continue to expand, the economic upper bound of a single interaction also rises accordingly. Under such conditions, any unnecessary expansion in reasoning length, retrieval scope, or execution trajectory can directly translate into higher API expenditure and a larger system-side computational burden. This makes resource consumption analysis increasingly important, not only for understanding attack surfaces but also for characterizing the practical cost environment in which such threats emerge.

D The Resource Consumption Threat Landscape

D.1 The Growing Pressure on Computational Resources

In LLM systems, computational cost is tightly coupled with the generation process itself: longer outputs require more autoregressive decoding steps (Vaswani et al., 2017) and impose growing sequence-length-dependent memory and compute overhead (Nayab et al., 2024; Wang et al., 2024b; Anagnostidis et al., 2023). As a result, resource pressure becomes an inherent property of modern generation systems. This relationship is already reflected in real-world deployment practice (Lodha, 2025), where providers charge separately for output tokens and offer discounted asynchronous batch services to better manage compute demand[2]. In large-scale deployments, such pressure can escalate into visible service risks; for example, the launch of DeepSeek-R1 attracted massive traffic, while the cost of long-form generation contributed to severe capacity strain and service restrictions[3]. These observations suggest that the generation process itself has become a bottleneck in modern model deployment, making resource-amplifying behaviors a critical concern for reliable and sustainable deployment.

D.2 Threat Scenarios Under Resource Consumption Regimes

Building on the taxonomy introduced in the main text, resource consumption threats can manifest under two representative regimes: ***Overthinking*** and ***Unbounded Drift***. Although both ultimately aim to amplify computational consumption during generation, they differ in the behavioral patterns of the generation process and in the types of risks they introduce.

Overthinking threats. In the Overthinking regime, the generation process remains semantically aligned with the task but becomes unnecessarily verbose or computationally intensive. Such behavior can arise when prompts or inputs induce the model to produce excessively detailed reasoning (Zhu et al., 2025a), redundant explanations (Kumar et al., 2025), or low-information-density de-

[2] https://openai.com/zh-Hans-CN/api/pricing/?utm_source=chatgpt.com

[3] https://www.reuters.com/technology/cybersecurity/deepseek-limits-registrations-due-cyber-attack-2025-01-27/?utm_source=chatgpt.com

Model	Input	Output
GPT-5.4 (<272K context length)[a]	$2.50	$15.00
GPT-5.4 (>272K context length)[a]	$5.00	$22.50
GPT-5.4-pro (<272K context length)[a]	$30.00	$180.00
GPT-5.4-pro (>272K context length)[a]	$60.00	$270.00
GPT-5 mini[a]	$0.25	$2.00
Claude Opus 4.6[b]	$5.00	$25.00
Claude Sonnet 4.6[b]	$3.00	$15.00
Claude Haiku 4.5[b]	$1.00	$5.00
Gemini 2.5 Pro (≤200K prompt)[c]	$1.25	$10.00
Gemini 2.5 Pro (>200K prompt)[c]	$2.50	$15.00
Gemini 2.5 Flash[c]	$0.15	$1.25
Gemini 2.5 Flash-Lite[c]	$0.10	$0.40
Gemini 2.0 Flash[c]	$0.05	$0.20
Mistral Large 2[d]	$2.00	$6.00
Mistral Medium 3 / 3.1[d]	$0.40	$2.00
Mistral Small[d]	$0.20	$0.60
DeepSeek-V3[e]	∼$0.27	∼$1.1
DeepSeek-R1[e]	∼$0.55	∼$2.2

[a] OpenAI pricing page: https://developers.openai.com/api/docs/pricing
[b] Anthropic pricing page: https://platform.claude.com/docs/en/about-claude/pricing
[c] Google AI pricing page: https://ai.google.dev/pricing
[d] Mistral pricing page: https://mistral.ai/pricing#api
[e] DeepSeek pricing page: https://api-docs.deepseek.com/quick_start/pricing/

Table 2: API pricing comparison of representative mainstream models. Prices are listed in USD per 1M input/output tokens.

Models	2020	2021	2022	2023	2024	2025
Mainstream Models	0.5K	2K	8K	32K	128K	128K
Frontier Models	1K	4K	16K	128K	512K	10M

Table 3: Context Window Comparison of Representative Mainstream Models

scriptions (Shumailov et al., 2021). This phenomenon is particularly prominent in reasoning-oriented models (Wei et al., 2022; Creswell and Shanahan, 2022; Hao et al., 2023; Guo et al., 2025a; He et al., 2025) and agent-based systems (Yao et al., 2022; Xi et al., 2025; Wang et al., 2024a; Chen et al., 2023a), where extended reasoning chains or repeated tool interactions may significantly expand the generation trajectory. For example, adversarial manipulations of reasoning prompts can induce models to generate prolonged chains of intermediate reasoning (Si et al., 2025), dramatically increasing token consumption and latency (Yi et al., 2025). Similar amplification effects have also been observed in multimodal systems, where carefully crafted inputs can trigger extremely verbose textual outputs despite containing limited semantic information (Gao et al., 2025). These behaviors may appear benign at the output level, yet they substantially inflate inference cost and degrade overall system throughput. When scaled across large numbers of queries, such amplification can accumulate into significant economic losses or operational overhead. Several realistic deployment scenarios illustrate how such behavior can be exploited in practice.

First, in commercial LLM services, a malicious or poorly regulated service provider could inten-

tionally design models or prompting pipelines that encourage unnecessarily verbose reasoning or redundant explanations, effectively increasing token consumption and inflating user-side API costs (Zhu et al., 2025b). Because users typically pay for output tokens (Alavi et al., 2025), even subtle increases in verbosity can translate into substantial financial overhead when deployed at scale (Sun et al., 2025; Lodha, 2025; Bergemann et al., 2025).

Second, in agent-based ecosystems where multiple tools, plugins, or intermediate services participate in a task pipeline, adversarial intermediaries may inject additional reasoning steps, auxiliary tasks, or low-value instructions into the agent workflow (Zhan et al., 2024; Zhao et al., 2025; Zhang et al., 2025c). Such manipulations can cause the model to perform extra computations that appear semantically related to the task while silently consuming additional API budget or computational resources (Dong et al., 2026; Wang et al., 2025f).

Third, overthinking behaviors can pose availability risks for resource-constrained deployments, such as small service providers or edge devices (Zheng et al., 2025; Reddi, 2025; Fung et al., 2025). In these settings, excessive generation may significantly delay task completion, block concurrent requests, or exhaust limited compute capacity, thereby reducing system responsiveness and degrading overall service reliability (Wang et al., 2023b; Chen et al., 2021).

Taken together, these scenarios suggest that seemingly benign verbosity in generation can become a practical vector for resource abuse when deployed in real-world LLM ecosystems.

Unbounded Drift, in contrast, refers to generation trajectories that progressively deviate from the intended task and fail to converge within a reasonable progression. Such behavior can arise when adversarial prompts (Gao et al., 2025), decoding perturbations (Geiping et al., 2024), or recursive reasoning (Wang et al., 2026b). This phenomenon is particularly prominent in autoregressive decoding and agentic systems (Wang et al., 2023a), where weakened termination signals, repetitive token dynamics, or recursive tool use may cause the generation trajectory to expand far beyond normal task completion. For example, carefully crafted malicious inputs can trap models in repetitive decoding loops or non-halting generation, forcing them to continue producing tokens until system-imposed limits intervene (Gao et al., 2025). Similar effects can also emerge in agent environments, where recursive tool-calling or multi-step interaction chains expand the execution trajectory far beyond the original task scope (Zhou et al., 2026; Lee et al., 2026). These behaviors are more overtly destructive than Overthinking, since they not only inflate computational cost but also directly threaten service availability and task completion. Several realistic deployment scenarios illustrate how such behavior can be exploited in practice.

First, malicious users or competing service providers may deliberately craft requests that suppress termination or trigger repetitive generation, forcing a provider to allocate disproportionate compute to a small number of adversarial queries (Gao et al., 2024c). When amplified at scale, such requests can monopolize shared resources, degrade service availability, and in extreme cases resemble denial-of-service attacks against generative LLM infrastructure (Li et al., 2025b; Wang et al., 2026a). Recent work explicitly frames this threat as inference-time DoS, where a small malicious input can monopolize GPU time, queue slots, or memory resources, starving legitimate users (Zhang et al., 2025e).

Second, untrusted intermediaries in user-facing pipelines may inject malicious suffixes (Zhao et al., 2025; Si et al., 2023), hidden instructions, or adversarial constructions into otherwise benign requests, causing the model to enter prolonged decoding trajectories or non-convergent loops. In this setting, the immediate consequence is reduced end-user usability, including longer wait times, unstable responsiveness, and higher usage costs. This threat is especially concerning because the injected content may remain invisible to end users while still shifting the request into a pathologically expensive generation regime.

Third, in agentic environments, adversarial intermediaries or compromised components may induce recursive tool-calling loops or self-amplifying interaction chains that both block task execution and corrupt task usefulness. Such behaviors not only consume excessive resources but may also trap the agent in stalled workflows, produce low-value or erroneous outputs, and obstruct downstream processes that depend on timely completion (Lee et al., 2026; Zhou et al., 2025). Recent studies show that tool-layer manipulation can expand agent trajectories to more than 60,000 tokens (Zhou et al., 2026), inflate costs by hundreds of times, and substantially reduce co-running throughput, while other attacks can covertly parasitize the user's compute

budget by injecting unauthorized auxiliary workloads into apparently legitimate workflows (Zhang et al., 2025b).

Taken together, these scenarios suggest that Unbounded Drift is not merely a generation abnormality, but a practical route through which resource abuse can escalate into service disruption, task failure, and system-level availability degradation.

D.3 Why Resource Consumption Security Matters

The preceding analysis highlights that resource consumption is no longer merely an efficiency issue but an emerging security concern in LLMs. Because modern model services operate on shared computational infrastructure, excessive generation or non-convergent interaction patterns can disproportionately consume limited resources and degrade service availability for other users. As LLMs continue to scale and become embedded in agentic workflows and real-world deployments, such behaviors may amplify operational costs, disrupt service reliability, and threaten the sustainability of model infrastructure. Despite these risks, existing studies remain fragmented across different models and system settings, lacking unified taxonomies. A clearer conceptual framework for understanding threats to resource consumption is therefore essential for advancing reliable and sustainable LLMs.

E A Unified View of the Survey Organization

To clarify the scope and internal structure of this survey, Figure 4 provides a unified overview of how the literature on resource consumption issues is organized in this work. Specifically, we structure the problem into two primary resource consumption regimes, *Overthinking* and *Unbounded Drift*, which serve as the main taxonomy throughout the paper. Building on this taxonomy, the survey is further developed from three complementary perspectives: **attacks**, which examine how resource amplification is induced in different systems; **mechanisms**, which analyze the underlying dynamics that lead to excessive or non-convergent generation; and **defenses**, which summarize current mitigation strategies and their coverage.

Within each perspective, existing studies are further grouped by system type, including large language models, reasoning models, multimodal large language models, and agentic systems. This organization is intended to provide a consistent view of the field: the taxonomy captures *what kinds* of resource consumption behaviors emerge, while the three research perspectives capture *how they are induced*, *why they arise*, and *how they may be mitigated*. In this way, the figure serves as a compact map of the survey, highlighting both the coverage of current research and the uneven maturity of different branches.

F Towards a Standardized Evaluation Framework for Resource Consumption Threats

Resource consumption threats are increasingly recognized as a critical safety concern in LLMs. While many studies have explored attacks and defenses across different modalities, model families, and deployment scenarios, the literature remains fragmented: evaluation metrics, datasets, and experimental setups vary widely, making cross-study comparison challenging. This appendix provides a structured overview of existing work and, based on observed gaps, proposes a unified evaluation framework to guide future research. The following sections first summarize the current landscape of attacks and defenses, then introduce recommended evaluation guidelines for standardized assessment.

F.1 Summary of Existing Work

Table 4 to Table 8 present representative resource consumption attacks, including the datasets used, evaluation metrics, targeted models, and observed efficiency or latency amplification. Table 9 and Table 10 summarize corresponding defenses and their reported effectiveness. These tables illustrate several key points:

High research activity across multiple modalities and settings: Existing studies cover text-based LLMs, reasoning models, multimodal systems, and agentic environments. Attacks have been demonstrated on a wide range of models, including GPT-family, Gemini, DeepSeek, and various perception-model pipelines.

Heterogeneous evaluation protocols: Different works report distinct metrics, such as token amplification, latency increase, attack success rate, or system-level impact. Similarly, datasets vary from standard benchmarks to real-world agent workloads, reflecting diverse experimental contexts.

Fragmented coverage and limited comparability: Most studies focus on specific attack types,

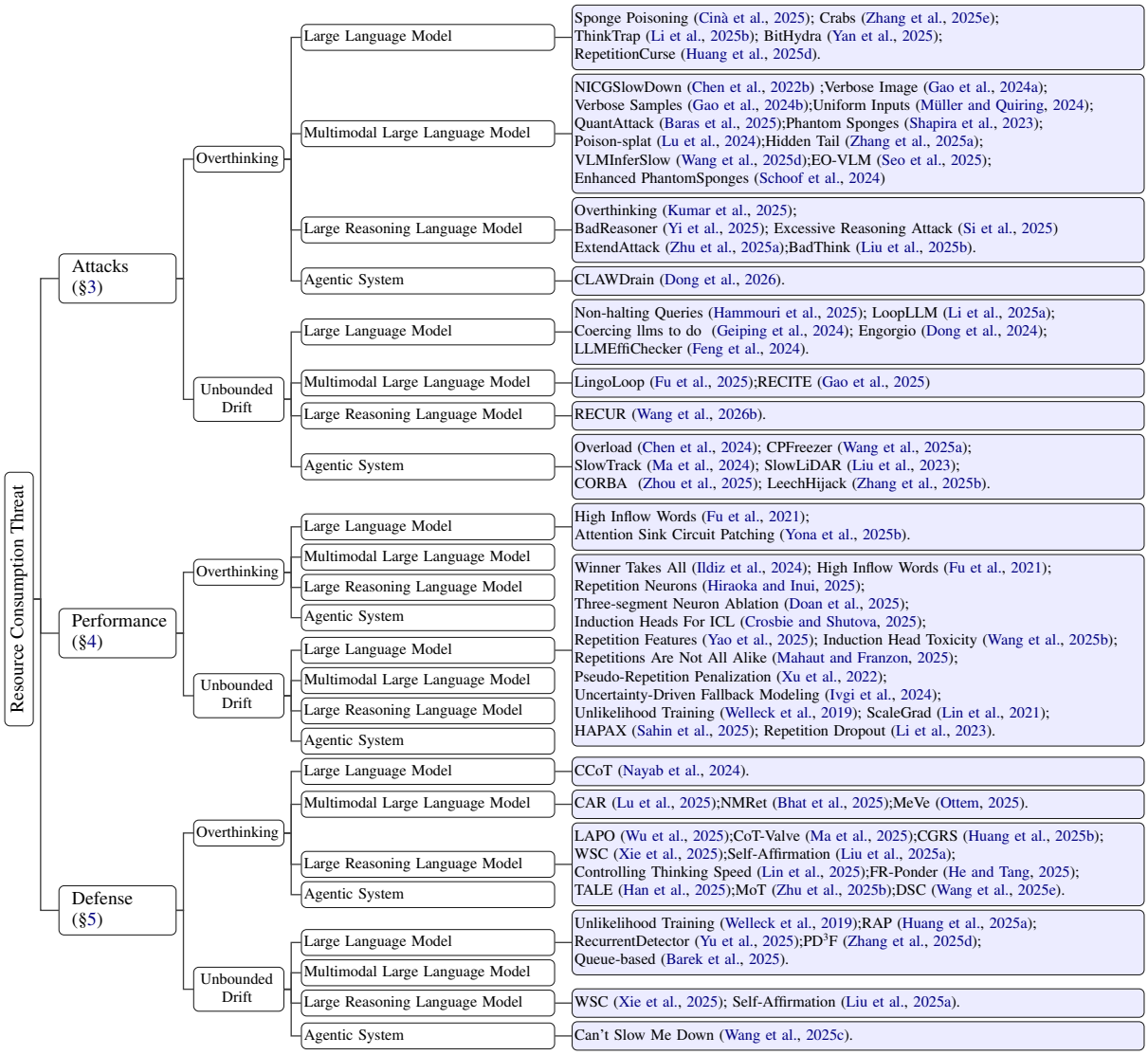

Figure 4: Overall organization of resource consumption issues across attack, mechanism, and defense perspectives.

model classes, or deployment scenarios. Direct comparison between methods is complicated by differing metrics, model sizes, and evaluation environments. While this demonstrates the growing interest in threats to resource consumption, it also highlights the need for consistent, standardized assessment.

These observations motivate the development of a unified evaluation framework that provides a common basis for comparing attacks and defenses and supports reproducible, comprehensive research.

F.2 Proposed Evaluation Guidelines

Given the heterogeneity of existing studies, we recommend evaluating resource consumption threats from three complementary perspectives: *model-level behavior*, *hardware-level pressure*, and *application-level impact*. This design follows the intuition that excessive generation first manifests as abnormal decoding behavior at the model level, then translates into measurable computational burden on the serving hardware, and finally affects the quality of service in real deployments. Evaluating all three levels together provides a more complete view of both attack severity and defense effectiveness.

At the *model level*, we recommend reporting **output length** and **inference latency**. Output length directly captures the extent to which generation is induced by the attack or controlled by the defense. This metric is particularly important because many resource consumption threats operate by triggering unnecessarily long outputs, repetitive reasoning traces, or non-terminating decoding patterns. It is also highly portable across model families, including standard LLMs, reasoning mod-

els, multimodal systems, and agentic components that generate intermediate textual outputs. However, output length alone is insufficient, since the same number of tokens may incur very different computational costs depending on the model architecture, decoding strategy, and modality. We therefore additionally recommend inference latency as a direct measure of time overhead. Latency reflects how long the system takes to process a query under attack or defense, and is especially useful for capturing practical degradation in responsiveness. Together, output length and latency characterize the immediate behavioral footprint of resource amplification at the model side.

At the *hardware level*, we recommend reporting **GPU utilization** and **memory utilization**. These metrics capture whether excessive generation is actually converted into pressure on the underlying compute infrastructure. GPU utilization measures how intensively the accelerator is occupied during inference, and is useful for identifying whether an attack induces sustained computational saturation rather than merely producing longer outputs. Memory utilization, including both allocated and peak memory usage where possible, reflects the storage burden created by long contexts, repeated decoding steps, or multimodal feature processing. This metric is especially important for large models and multimodal systems, where memory bottlenecks may become the dominant constraint even before compute is fully saturated. Compared with model-level metrics, hardware-level metrics are more deployment-sensitive, since they depend on serving frameworks, batching policies, quantization, and device types. Nevertheless, they are essential when the goal is to assess the actual infrastructure impact of a threat or the practical efficiency gain of a defense.

At the *application level*, we recommend reporting **throughput**. Throughput reflects the number of requests or tasks that can be completed within a unit of time, and therefore serves as a direct indicator of service capacity under attack or defense. This metric is particularly important because the ultimate consequence of resource consumption threats is often not limited to a single query becoming slower, but rather to the whole system handling fewer users, fewer concurrent tasks, or fewer agent executions. Throughput thus captures the service-level manifestation of resource abuse, making it especially relevant for online APIs, multi-user serving platforms, and agentic systems with concurrent workflows. In practice, throughput should be interpreted jointly with latency, since a system may maintain acceptable latency for individual queries while still suffering degraded overall capacity under increased workload.

Taken together, these metrics form a layered evaluation framework. Output length and inference latency capture how the model behaves under resource amplification; GPU and memory utilization measure whether this behavior imposes real hardware burden; and throughput reflects whether such a burden ultimately degrades service capacity in practical deployments. This layered design also improves comparability across settings. Model-level metrics are broadly applicable and should be reported in nearly all studies. Hardware-level metrics are particularly important for systems and deployment-oriented evaluations. Application-level metrics are most informative in realistic serving scenarios, especially for production APIs and agentic environments. We therefore recommend that future studies report at least one metric from each level whenever possible, so that resource consumption threats can be assessed not only as generation anomalies, but also as infrastructure and service risks.

G Detailed Technical Analysis of Resource Consumption Mechanisms

Rather than reiterating the high-level taxonomy presented in the main text, this section provides a **more fine-grained technical account** of the mechanisms underlying resource consumption behaviors in autoregressive generation. We organize the discussion across four complementary levels of analysis—Markov chain foundations, circuit-level mechanisms, behavioral dynamics, and training-level causes—so that researchers can more precisely trace how repetition and output prolongation arise from theoretical properties, internal model components, emergent generation patterns, and learning objectives, respectively.

G.1 Markov Chain

By modeling autoregressive generation as a Markov chain and deriving theoretical upper bounds on the Average Repetition Probability (ARP), (Fu et al., 2021) identifies *High Inflow Words*—tokens toward which disproportionately many others transition with high probability. These tokens form absorbing-like loops within the tran-

sition matrix: once the generative process enters such high-inflow states, it becomes trapped in repetitive cycles rather than progressing toward the EOS token. The proposed rebalanced encoding merges high inflow pairs into single tokens, effectively reducing the inflow term in the ARP bound and significantly lowering repetition rates in both translation and language modeling tasks. From a complementary angle, (Ildiz et al., 2024) establishes a formal equivalence between single-layer self-attention and Context-Conditioned Markov Chains (CCMC), and analyzes the autoregressive trajectory from a single prompt. Due to the non-mixing nature of CCMC, majority tokens undergo self-reinforcement across generation steps, causing the output distribution to collapse into a singleton or highly limited token subset—providing a principled mathematical account of why LLMs tend to generate repetitive text during prolonged decoding.

G.2 Circuit-Level Mechanisms

Repetition Neurons and Features Specific neurons within MLP blocks have been identified as direct executors of repetitive content. Differential activation analysis (Hiraoka and Inui, 2025) revealed "repetition neurons" distributed throughout the model: intermediate-layer neurons tend to detect repeating patterns, while top-layer neurons drive the model to replicate previous context. Layer-wise causal analysis (Doan et al., 2025) further shows that these neurons often function as downstream components of Induction Heads, serving to amplify copying signals. Moving beyond individual neurons, (Yao et al., 2025) utilized Sparse Autoencoders (SAEs) to identify latent directions that specifically encode "repetition features," indicating that the model explicitly represents repetitive behaviors within its feature space.

Induction Heads Through prefix matching score analysis and targeted attention knockouts, (Crosbie and Shutova, 2025) provides quantitative evidence that induction heads are the fundamental operators of few-shot in-context learning (ICL). Because output repetition functions as an uncontrolled, degenerate form of ICL, Induction Head Descaling (Wang et al., 2025b) formalized the "Induction Head Toxicity" theory to explain this dynamic. From a mechanistic perspective, toxicity occurs when induction heads disproportionately dominate the output logits, thereby suppressing contributions from other attention heads. This logit dominance enforces rigid pattern replication and triggers a rapid entropy collapse in the next-token probability distribution. Adding structural nuance to this, (Mahaut and Franzon, 2025) revealed that repetition is actually sustained by two distinct, parallel mechanisms. The first is the aforementioned ICL-induced repetition, which relies on a specialized, late-developing circuit of attention heads and MLPs that operate with high prediction confidence. The second is a naturally occurring repetition mechanism that emerges very early in training; rather than using a dedicated circuit, it functions as a degenerate fallback, with attention weights anomalously collapsing onto low-information, structural tokens (e.g., newlines), sustaining repetitive loops even without explicit contextual prompts.

G.3 Behavioral Dynamics

By manually constructing pseudo-repetitive data and comparing token probabilities across increasing repetition counts, Pseudo-Repetition Penalization (Xu et al., 2022) reveals a critical *self-reinforcement effect*: once a model generates one repeated sentence, the probability of continuing to repeat rises almost monotonically with the number of historical repetitions, eventually stabilizing at a high ceiling value. Sentences with higher initial probabilities exhibit stronger self-reinforcement, explaining why maximization-based decoding is particularly prone to sentence-level loops. Based on this finding, the proposed DITTO method trains models to exponentially decay repetition probability on pseudo data, significantly reducing repetitions without sacrificing perplexity. From a complementary angle, uncertainty-driven fallback modeling (Ivgi et al., 2024) systematically varies model size, pretraining tokens, and instruction tuning to analyze fallback behaviors under *epistemic uncertainty*. Their experiments reveal a consistent ordering: as uncertainty increases during generation, models shift from producing correct facts to hallucinations, then to degenerate text, and finally to verbatim sequence repetitions—positioning repetition as the simplest fallback state when parametric knowledge is exhausted.

G.4 Training-Level Causes

Unlikelihood training introduces auxiliary losses to penalize repeated tokens (Welleck et al., 2019). Experimental results demonstrate that the approach significantly reduces repetition and dullness while maintaining competitive perplexity and token ac-

curacy, and produces higher-quality generations under standard decoding strategies such as greedy search and beam search. However, such losses alone do not resolve the skewed token-level probabilities inherent in MLE—motivating direct gradient manipulation as an alternative (Lin et al., 2021). ScaleGrad (Lin et al., 2021) directly modifies the gradients of the training objective to encourage the model to assign higher importance to novel tokens during learning. The approach effectively reduces repetition and enhances diversity while maintaining strong performance, as evidenced by both automatic metrics and human evaluations. A more targeted approach, HAPAX (Sahin et al., 2025), omits the loss contribution of any token predictable by induction heads—effectively ensuring that repeated n-grams within a context window never produce gradient signals. Despite a 66% reduction in verbatim copying, HAPAX models surpass the vanilla baseline on 13 of 21 abstractive in-context learning tasks, demonstrating that suppressing inductive copying does not compromise broader ICL capabilities. At the data level, Repetition Dropout (Li et al., 2023) randomly drops attention to repetitive words during training, directly reducing the model's exposure to repeated patterns. This simple strategy substantially lowers the repetition rate in generated text, and further analysis shows that it provides a unified explanation for prior methods—penalizing training-data repetitions emerges as the common and fundamental factor underlying the effectiveness of high-inflow word mitigation, likelihood objective modifications, and self-reinforcement suppression. Notably, this effect persists across larger model scales and instruction-tuned settings.

H Detailed Discussion of Mitigation Methods

Mitigation is a necessary counterpart to the study of resource consumption threats, especially as recent attacks have shown that resource abuse can escalate from efficiency degradation to severe operational and economic harm. The main text has already provided a high-level overview of the current defense landscape, focusing on the available mitigation strategies, their coverage, and the major gaps that remain. Rather than reiterating the high-level taxonomy in the main text, this appendix provides a **more fine-grained technical account** of representative defense methods, so that future researchers and practitioners can more easily identify the most suitable interventions for different threat regimes and deployment settings. Specifically, we expand the discussion along the same two primary threat classes identified earlier—*Overthinking* and *Unbounded Drift*—and further organize existing methods by the system layer at which intervention occurs, including training, decoding, and external control, with emphasis on their **technical mechanisms**, **reported effects**, and practical limitations.

H.1 Defenses against Overthinking

Large Language Models (LLMs). This line of work mitigates overthinking primarily through *prompt-level output control*, where conciseness constraints are explicitly injected into the input to suppress unnecessary reasoning expansion. Constrained-CoT (CCoT) (Nayab et al., 2024) implements this idea by introducing user-specified length budgets (e.g., 15, 30, or 45 words) directly into the prompt, forcing the model to trade off answer correctness against output brevity during decoding. To evaluate this trade-off, it defines three correctness metrics: Hard-k Concise Accuracy (HCA), which counts only correct responses within a fixed length threshold; Soft-k Concise Accuracy (SCA), which applies an exponential penalty to moderate length violations; and Consistent Concise Accuracy (CCA), which further measures the stability of concise reasoning across generations. It also introduces the Redundancy Mean Score (RMS) and the Information Flow Score to quantify syntactic redundancy and semantic continuity in generated reasoning traces. In essence, these methods do not modify model parameters or decoding dynamics, but instead rely on external prompt-side constraints to compress generation, which makes them lightweight yet inherently limited against adversarially induced length amplification.

Reasoning Large Language Models (RLLMs). In reasoning models, a more direct line of defense is to *internalize length control during training*, so that budget awareness becomes an intrinsic capability of the model rather than an external prompt-side constraint. LAPO (Wu et al., 2025) implements this idea with a two-stage reinforcement learning procedure. In the first stage, the model performs GRPO rollouts and records the output lengths of correct responses only; the 30th and 70th percentiles are then used to define a filtered target interval $[L_{\min}, L_{\max}]$, and a linear decay reward penalizes generations that fall outside this range. The median feasible

length L_{median} is then carried into the second stage as an explicit self-budgeting target embedded in the prompt, while a Gaussian-style length-adherence reward encourages the model to match its declared budget during generation. This effectively converts length control from post hoc truncation to learned planning behavior, yielding up to 40.9% token reduction and a 2.3% accuracy gain on mathematical reasoning benchmarks.

CoT-Valve (Ma et al., 2025) addresses the same problem through *parameter-space controllability* rather than reward shaping. Using LoRA, it identifies a controllable direction in parameter space that governs reasoning length, allowing a single model to compress or expand its chain of thought by adjusting the intervention magnitude. To support this control, the method constructs the MixChain dataset, where each question is paired with reasoning traces of different lengths. Under this design, reasoning compression is achieved as a continuous model-side capability rather than a fixed decoding heuristic: on QwQ-32B, the method reduces GSM8K reasoning length from 741 to 225 tokens with only a 0.15% accuracy drop, and compresses AIME traces from 6,827 to 4,629 tokens with only one additional error.

At the decoding stage, existing defenses mitigate overthinking through *lightweight inference-time control* without updating model parameters. A first line of work suppresses explicit reflection triggers once continued reasoning becomes unnecessary. CGRS (Huang et al., 2025b), for example, inserts a probe at structural delimiters and estimates current confidence from output entropy; when the confidence exceeds 0.9, it directly downweights reflection-leading words such as "Wait", "But", and "Alternatively" by assigning them large negative logits, thereby preventing further reflection loops at decoding time.

A second line of work detects and interrupts semantically empty repetition from internal representations. WSC (Xie et al., 2025) targets useless self-repetitions that consume decoding budget without adding semantic value. It trains a model-specific linear classifier on hidden states at end-of-chunk positions and triggers chopping when either 2 consecutive long repetitive chunks ($\geq$10 tokens) or 5 short ones ($<$10 tokens) are detected; after interruption, a rescue regeneration prompt such as "Let me reconsider..." is appended under a fixed token budget. The self-affirmation suppression approach (Liu et al., 2025a) focuses on a narrower redundancy mode, namely reflective steps that merely reaffirm earlier correct content, and suppresses them by exploiting probability biases in their leading words, achieving length reductions of 18.7% in the train-free setting and 50.2% in the train-based setting.

A third line of work performs *representation-level steering* of reasoning depth. Controlling Thinking Speed (Lin et al., 2025) extracts a steering vector by applying PCA to hidden-state difference vectors between fast- and slow-thinking trajectories, and then adjusts reasoning speed through a strength parameter α ($\alpha > 0$ accelerates thinking and $\alpha < 0$ slows it down). It further introduces an adaptive variant that estimates real-time reasoning difficulty from the Jensen–Shannon divergence between early-layer and final-layer logits, enabling dynamic switching within a single inference pass and yielding an average $+1.3\%$ accuracy gain together with a -8.6% token reduction. FR-Ponder (He and Tang, 2025) adopts a related latent-control strategy, extracting contrastive steering vectors from step-by-step reasoning versus direct-answer prompts and applying additive hidden-state perturbations through a lightweight controller trained with GRPO, with curriculum learning used to align compute allocation to task difficulty; under this design, it reports 30–40% token reduction with up to 10% accuracy improvement on GSM8K, MATH500, and GPQA.

A fourth line of work controls overthinking by allocating *sampling-time budgets*. DSC (Wang et al., 2025e) first performs batch-level difficulty ranking using the model itself: queries with zero entropy across pre-samples are assigned to an "Easy" group and receive only a single chain-of-thought sample, whereas "Hard" queries are assigned an initial sampling budget estimated from similarly difficult prior queries. If consensus is not reached, the sampling window is further expanded using a Dirichlet-based stopping rule. This makes DSC fundamentally a batch-level budget scheduler rather than a single-query real-time controller, which limits its applicability in latency-sensitive inference settings.

From an external control perspective, existing methods mitigate overthinking by imposing *budget signals or supervisory constraints outside the model itself*, without directly modifying the underlying reasoning dynamics. TALE (Han et al., 2025) implements this idea through token-budget control conditioned on problem complexity: TALE-EP estimates an appropriate budget via zero-shot prompting, while TALE-PT internalizes budget awareness

through SFT/DPO post-training, so that the budget can be explicitly injected into the reasoning process as an external control variable. In a simpler form, CCoT (Nayab et al., 2024) applies the same prompt-side control principle by directly embedding length constraints into user instructions, thereby capping output verbosity without changing model parameters or decoding rules.

A more intervention-oriented line of work introduces *external supervisors* that monitor and terminate redundant reasoning online. MoT (Zhu et al., 2025b) follows this design by exploiting the manipulability of special delimiter tokens, the same external control surface used in adversarial BoT-style attacks, to insert a plug-and-play monitoring mechanism over the reasoning process. Rather than compressing reasoning through static prompting alone, it dynamically halts redundant or risky reasoning paths during generation, serving both as an efficiency controller to reduce overthinking and as a safety-oriented supervisor to terminate unsafe reasoning, with a monitoring interval of every 200 tokens.

Multimodal Large Language Models (MLLMs) and Memory Systems. In multimodal and long-context settings, overthinking mitigation must address not only unnecessary reasoning depth, but also the growth of retrieved or cached context. A first line of work controls *whether long-form reasoning is invoked at all*. CAR (Lu et al., 2025) implements this through perplexity-based routing, dynamically switching between short responses and chain-of-thought reasoning according to the model's estimated confidence, so that expensive long-form reasoning is triggered only for uncertain inputs.

A second line of work controls *how much contextual state is carried into generation*. NM-Ret (Bhat et al., 2025) manages long-context overhead through a structured memory architecture consisting of a stateful neural memory for abstract long-term storage, a vector store for contextual retrieval, and a reasoning compressor for intermediate context management, thereby reducing context-window growth during multi-step reasoning. MeVe (Ottem, 2025) approaches the same problem from the retrieval side by inserting explicit verification and compression before generation: it performs initial kNN retrieval, cross-encoder-based relevance filtering, BM25 fallback retrieval, relevance-based context prioritization with redundancy removal, and finally token-budgeted greedy packing to control the final context size. Under this design, irrelevant or low-value context is filtered before entering the model, yielding a 57% reduction in context token consumption on Wikipedia datasets and 75% on HotpotQA.

H.2 Defenses against Unbounded Drift

The following methods target non-convergent generation, where repetition collapse, infinite loops, or runaway execution amplify resource consumption beyond output inefficiency and into crash-level system pressure.

Large Language Models (LLMs). For non-convergent generation in general LLMs, existing defenses mainly intervene at the training and decoding stages. At training time, the core idea is to reshape the learning objective so that repetitive patterns are explicitly suppressed during optimization. Unlikelihood training (Welleck et al., 2019) implements this by introducing negative updates at two granularities. At the token level, previously generated tokens in the context are treated as negative candidates, and an unlikelihood term is added to the standard MLE objective to penalize high probability assigned to those historical tokens, thereby suppressing local repetition and high-frequency token dominance. At the sequence level, tokens belonging to repeated n-grams in the decoded outputs are marked as negative samples during fine-tuning, extending this principle to longer structural repetition patterns. These two objectives are combined in training, though they exhibit different limitations: token-level optimization suffers from a distribution mismatch between training and generation, whereas sequence-level optimization requires decoding full sequences within the training loop and is therefore substantially more expensive. Under beam search, the combined design reduces the 4-gram repetition rate from 0.442 to 0.013 and increases the number of unique generated tokens by 77%.

At decoding time, the focus shifts from suppressing repetition in the learned distribution to controlling it online during generation. RAP (Huang et al., 2025a) formalizes this process through systematic tuning of the Repetition Penalty Parameter (RPP). Its core measurement component, ReDA, computes a repetition ratio (RR) for each output sequence by detecting not only standard textual repetition but also consecutive non-word-character rep-

etition and space-free long-form repetition through regular-expression-based matching. Given RR, RAP selects the penalty strength by maximizing the score $P \times F(\mathrm{RR})$, where P denotes task performance and $F(\mathrm{RR})$ is a penalty function over repetition severity. Among five candidate forms—linear, quadratic, cubic, logarithmic, and exponential—the cubic function $(1 - \mathrm{RR})^3$ yields the best trade-off according to the reported ablation results.

For more severe resource exhaustion and DoS-style threats, the most mature defenses operate at the *system level*, where intervention is applied during serving rather than through model retraining. A representative direction is *online loop detection*. RecurrentDetector (Yu et al., 2025) monitors Transformer activation states at each generation step with a lightweight MLP classifier and terminates generation once the cosine similarity between the current and previous states exceeds 0.95, treating high state recurrence as a signal of non-convergent looping. Under this design, it achieves 95.24% detection accuracy with a 2.59% false positive rate and only 0.36ms additional latency. Its scope, however, is limited by strong white-box assumptions: it cannot be applied to closed-source APIs such as GPT-4, may be bypassed by polymorphic attacks that preserve semantic loop structure while varying surface tokens, and has been evaluated on only 6 open-source architectures across 4,000 prompts.

A complementary direction performs *resource-aware serving control* under adversarial load. PD^{3}F (Zhang et al., 2025d) implements this through a two-stage pipeline. On the input side, it computes a Resource Index from five features — total inference time, peak GPU memory, peak GPU utilization, input length, and output length — and uses this signal to guide dynamic request polling under high-concurrency, malicious prompts. On the output side, it applies an Adaptive End-Based Suppression mechanism that explicitly amplifies the EOS logit to terminate maliciously prolonged generation. Under AutoDoS-style attacks, this design improves legitimate-user throughput by up to 500%, achieves attack-detection accuracy above 99% across AutoDoS, GCG-DoS, and P-DoS settings, and has been validated across six open-source LLMs. Its main limitation is threshold sensitivity: the IQR-based decision rule may misclassify legitimate but computationally intensive requests, such as long code generation, leading to substantial false positives in real deployments.

At a broader infrastructure level, queue-based web service architectures (Barek et al., 2025) mitigate overload by *decoupling request admission from direct model execution*. Instead of treating every request as an immediately executable generation job, they use distributed queuing to smooth bursty demand, isolate overload pressure from the serving backend, and preserve stable, near-linear scalability under extreme workloads. Compared with model-centric defenses, this line of work does not directly suppress malicious generation behavior but provides an engineering-level containment mechanism for maintaining service stability when resource abuse cannot be fully blocked upstream.

Reasoning Large Language Models (LRMs). Purpose-built defenses against crash-level and non-terminating behavior in reasoning models remain largely absent. Existing mitigation methods only provide *incidental coverage* by suppressing semantically empty self-repetition during overthinking control. For example, the self-affirmation suppression approach (Liu et al., 2025a) and WSC (Xie et al., 2025) can interrupt mild repetitive loops because both target redundant reflective steps that consume decoding budget without adding semantic value. However, these methods are not designed to handle adversarially induced collapse. WSC relies on a model-specific classifier, and its regeneration stage may itself enter a loop, while CGRS depends on hardcoded reflection triggers and can therefore be bypassed when prompt injection suppresses or substitutes those exact tokens. As a result, current reasoning-model defenses do not yet provide reliable protection against sustained crash-style attacks, leaving a clear gap between overthinking mitigation and true non-termination defense.

Multimodal Large Language Models (MLLMs). For latency-oriented resource threats in multimodal and edge-deployed systems, existing defenses mainly intervene through *hardware-aware adversarial training*. Background-attentive adversarial training (Wang et al., 2025c) implements this idea by explicitly coupling perturbation robustness with device-level capacity constraints across heterogeneous GPUs. Technically, it uses binary masks to concentrate perturbation awareness on vulnerable background regions, where latency attacks often induce excessive post-processing overhead, and incorporates objectness loss as an auxiliary signal to distinguish true objects from attacker-induced phantom detections. Under this design, the defense improves both robustness and serving efficiency:

on Jetson Orin NX, it restores processing speed from 13 FPS to 43 FPS, achieves 8–10% higher robust accuracy than MTD and OOD, and incurs only 4.4% clean accuracy loss. The reported evaluation covers multiple YOLO-based detectors, including YOLOv3, YOLOv5, and YOLOv8, across autonomous driving and general object detection settings.

I Benchmarks and Datasets

Existing benchmarks for resource consumption in large language models remain relatively limited. A small number of benchmarks explicitly focus on excessive generation and DoS–style resource consumption. For example, BenchOverflow (Feiglin et al., 2026) measures the overflow phenomenon where benign plain-text prompts trigger abnormally long outputs. Prompt-Induced Over-Generation as Denial-of-Service (Guo et al., 2025b) constructs an attack-side benchmark that evaluates how prompts can induce over-generation under black-box access.

In contrast, a larger body of benchmarks focuses on efficiency issues arising from long-chain reasoning. Stop Overthinking (Sui et al., 2025) surveys efficient reasoning methods and highlights the prevalence of excessive reasoning behavior in chain-of-thought generation. EffiReasonBench (Huang et al., 2025c) proposes a unified benchmark for evaluating efficiency–performance trade-offs across multiple reasoning tasks and models. SafeChain (Jiang et al., 2025) studies the safety implications of long chain-of-thought reasoning and introduces datasets for evaluating safety and robustness in reasoning-intensive settings.

Overall, while recent work has begun to establish evaluation frameworks for both resource-consumption attacks and reasoning efficiency, existing benchmarks remain fragmented across different research objectives. A unified evaluation paradigm for resource-aware generation and robustness is still largely lacking.

J AI Writing Assistance Disclosure

We used AI tools solely for language polishing to improve clarity and readability. The AI tools did not contribute to the scientific content, ideas, analyses, or conclusions of this work.

Method	Dataset	Evaluation Metrics	Models	Efficiency
A sloth (Multi-exit) (Hong et al., 2020)	CIFAR-10, ImageNet, Tiny-ImageNet	Exit Index, Avg Latency, Energy	MSDNet, ResNet (Multi-exit)	N/A
SkipSponge (Lintelo et al., 2024)	CIFAR-10, SVHN	FLOPs Inflation, Accuracy	ResNet, WideResNet	N/A
On-Device Sponge (Wang et al., 2023b)	Speech Commands, IMDB	Battery Drain, Execution Time	MobileNetV2, ShuffleNet	N/A
Dynamic Routing (Chen et al., 2023b)	CIFAR-100, ImageNet	Avg Layers, Activation Ratio	GaterNet, SkipNet	N/A
Sponge Poisoning (Wang et al., 2023b)	ImageNet, CIFAR	Latency, GPU Energy	VGG, ResNet	N/A
Energy Backdoor (Meftah et al., 2025)	GLUE, SQuAD	Energy Increase, Clean Accuracy	BERT, RoBERTa	512*
Sensing AI Sponge (Hasan et al., 2025a)	Audio/Sensor Data	Sensor Power, Pruning Ratio	CNN, DeepSense	N/A
Transslowdown (Chen et al., 2021)	WMT'14 (En-De), WMT'16 (En-Ro)	Translation Latency, Token Count, BLEU Score	Transformer, LSTM-based NMT	1024
NMTSloth (Chen et al., 2022a)	WMT'17 (En-De), WMT'19 (Zh-En)	Real-world Latency, Energy, Response Time	Fairseq, OpenNMT, MarianNMT	**80x**

Table 4: Summary of representative resource consumption attacks in early adaptive architectures.

Method	Dataset	Evaluation Metrics	Models	Efficiency
AutoDoS (Zhang et al., 2025e)	Chatdoctor, MMLU, Hellaswag, Codexglue, GSM	Attack Success Rate, Safety Compliance Rate, Task Success, Token Usage, Latency, Compute Cost	GPT, Llama, Qwen, Deepseek, Ministral, Gamma	8192
Repeated Token (Yona et al., 2025a)	OpenWebText, Custom Repetition Dataset	Attention Scores, Loss, Output Length	Llama-3, GPT-2, Pythia	8192
Non-halting Queries (Hammouri et al., 2025)	RAG Systems, 10k Probe Dataset	Avg Token Length, Generation Success	GPT-4o, Llama-3, Gemma-2	8192
ThinkTrap (Li et al., 2025b)	Sponge, LLMEffiChecker	Throughput, Response Latency, ASR	GPT-4o, Gemini 2.5 Pro, DeepSeek R1	8192
Crabs (AutoDoS) (Zhang et al., 2025e)	MMLU, GSM8K, Chatdoctor, Codexglue	Latency Inflation, Output Length ↑, GPU Memory	GPT-4, Llama-3, Qwen-2, DeepSeek, Gamma	8192
BitHydra (Yan et al., 2025)	MMLU, GSM8K	Inference Cost, Bit-flip Rate, ASR	Llama-2, OPT, Bloom	8192
Coercing LLMs (Geiping et al., 2024)	AdvBench, Vicuna-bench	ASR, Safety, Efficiency Loss	Vicuna, Llama-2, GPT-3.5	4096
Engorgio Prompt (Dong et al., 2024)	ShareGPT, Alpaca	Response Length, Energy Draw	Llama-3, Mistral, GPT-4	8192
LLMEffiChecker (Feng et al., 2024)	Efficiency-Bench, WikiText	Delay, Energy, Throughput	Llama-2, Vicuna, Alpaca	8192
LoopLLM (Li et al., 2025a)	MT-Bench, Chatbot Arena	Repetition Ratio, Energy Latency	Llama-3, Qwen-2, Phi-3	8192
DoS Poisoning (Gao et al., 2024c)	Anthropic HH-RLHF	Inference Latency, Poisoning Ratio	GPT-Neo, RoBERTa	1024

Table 5: Summary of representative resource consumption attacks in LLMs.

Method	Dataset	Evaluation Metrics	Models	Efficiency
Hidden Tail (Zhang et al., 2025a)	MS-COCO	Attack Success Rate,Output Length, Latency, Visible length, Response Quality	Qwen2.5-VL, MiMo-VL-7B-RL,Gemma-3-4B-IT	1831
NICGSlowDown (Chen et al., 2022b)	MS-COCO, Flickr8k	I-Loops, I-Latency (CPU/GPU)	ResNext-LSTM, GoogLeNet-RNN, MobileNets-LSTM	N/A
Verbose Images (Gao et al., 2024a)	MS-COCO, ImageNet	Sequence Length, Energy, Latency, Uncertainty	BLIP,BLIP-2, InstructBLIP, MiniGPT-4	8x
Verbose Samples (Gao et al., 2024b)	MSVD, TGIF	Length, Latency, Energy	VideoChat-2,Video-Vicuna,Video-LLaMA	4x
Uniform Inputs (Müller and Quiring, 2024)	ImageNet,	Activation Density, Activation Sparsity	ResNet, DenseNet, MobileNetV2	N/A
LingoLoop (Fu et al., 2025)	MS-COCO, ImageNet	Tokens, Energy Latency	InstructBLIP, Qwen2.5-VL, InternVL3	2048
Phantom Sponges (Shapira et al., 2023)	Berkeley Deep Drive(BDD), Mapillary Traffic Sign Dataset (MTSD), LISA, PASCAL VOC	Number of candidates, Processing Time, Detection Recall	YOLOv5, YOLOv3, YOLOv4	N/A
Enhanced PhantomSponges (Schoof et al., 2024)	Berkeley Deep-Drive	Number of candidates, Processing Time, Detection Recall	YOLOv5	N/A
RECITE (Gao et al., 2025)	ImageNet, MMLU, HumanEval, GSM8K	Attack Success Rate, Service response latency, GPU utilization, Average generation length	InstructBLIP, LLaVA, Qwen-VL	2048
EO-VLM (Seo et al., 2025)	-	GPU Power, Inference Time	YOLOv8, MASKDINO, Detectron2	N/A
QuantAttack (Baras et al., 2025)	ImageNet	GPU Memory, Processing Time, Energy, Outlier Count, Accuracy	Vision Transformer (ViT), Data-efficient image Transformer (DeiT)	N/A
VLMInferSlow (Wang et al., 2025d)	MS-COCO, ImageNet	I-length I-latency I-energy	FLAMINGO, BLIP, GIT, FLORENCE	N/A
Sponge Examples (Shumailov et al., 2021)	CIFAR-10, SVHN, WikiText-103, En-De	Energy Consumption, Latency, Memory Access	VGG-16, ResNet-18, GPT-2	512

Table 6: Summary of representative resource consumption attacks in MLLMs.

Method	Dataset	Evaluation Metrics	Models	Efficiency
OverThink (Kumar et al., 2025)	FreshQA, SQuAD, MuSR	Reasoning token amplification, Answer correctness, Guardrail evasion, Defense evaluation	o1, o1-mini, DeepSeek-R1	46x
Excessive Reasoning Attack (Si et al., 2025)	GSM8K, ORCA	Reasoning length and output length, utility performance (accuracy)	DeepSeek-R1-Distill-LLaMA, DeepSeek-R1-Distill-Qwen, o1-mini, o3-mini, DeepSeek-R1, QWQ	9x
BadReasoner (Yi et al., 2025)	-	Reasoning verbosity, answer correctness, controllability	Marco-o1, QwQ, DeepSeek-R1	N/A
BadThink (Liu et al., 2025b)	MATH-500, GSM8K	ASR, RIR, TAC, BAD, SD	DeepSeek-R1-Distill-Qwen (1.5B/7B/14B/32B), OpenR1-Qwen-7B, Light-R1-7B-DS	63.85x
ExtendAttack (Zhu et al., 2025a)	AIME 2024, AIME 2025, HumanEval, BigCodeBench-Complete	Response Length, Latency, Accuracy(Pass@1)	o3, o3-mini, QwQ-32B, Qwen3-32B	N/A
RepetitionCurse (Huang et al., 2025d)	-	prefill latency,TTFT,router imbalance	Mixtral-8x7B series, Qwen3-30B-A3B series, GPT-OSS-20B/120B, Kimi-Linear-Instruct, DeepSeek-V2-Lite, Llama-4-Scout-17B-16E-Instruct	N/A

Table 7: Summary of representative resource consumption attacks in RLLMs.

Method	Dataset	Evaluation Metrics	Models	Efficiency
SlowLiDAR (Liu et al., 2023)	KITTI, nuScenes	Runtime latency, Imperceptibility	PointPillars, SECOND, PV-RCNN	2.7x
CORBA (Zhou et al., 2025)	LLM-MAS simulation tasks	P-ASR, PTN, Availability degradation	GPT-4o-mini, GPT-4, GPT-3.5-turbo, Gemini-2.0-Flash, Qwen2.5-14B-Instruct, Llama-3.1-70B-Instruct, Gemma-2-27B-it	N/A
CP-FREEZER (Wang et al., 2025a)	OPV2V	End-to-end latency, Attack success rate, Frame processing time	OpenCOOD	90x
SlowTrack (Ma et al., 2024)	MOT17	Latency increase (R-Lat), Imperceptibility, System-level crash rate	SORT, FairMOT, ByteTrack, BoT-SORTSORT (YOLOv5 detector), FairMOT, ByteTrack, BoT-SORT	4x
LeechHijack (Zhang et al., 2025b)	GAIA, GPQA, MMLU	Attack success rate, Resource overhead, Detectability (stealthiness)	DeepSeek, Qwen, GPT, Gemini	N/A
Overload (Chen et al., 2024)	MS COCO dataset	Inference time, NMS latency	YOLOv5	10x
Clawdrain (Dong et al., 2026)	Real OpenClaw agent workloads	Token amplification, Cost overhead, Attack success rate, Stealthiness	OpenClaw v2026.2.9	∼9x

Table 8: Summary of representative resource consumption attacks in Agents.

Method	Dataset	Evaluation Metrics	Models	Efficiency	Model Class
PD^{3}F (Zhang et al., 2025d)	MMLU, Hellaswag, HumanEval, GSM, GPQA	Attack Success Rate, Task Success, Token Usage, Latency	Llama, Mistral, Qwen	Total Time 50% ↓	LLM
RecurrentDetector (Yu et al., 2025)	Custom trigger dataset (2388 inputs), ShareGPT	Accuracy, F1, FPR, Recall, trigger attempts, latency	Llama-3/2 (7/13B), Vicuna-v1.5 (7/13B), Gemma-2 (2B), GPT-4o/mini	RecurrentGenerator: avg attempts 272.1 vs 1679.1 random; RecurrentDetector: 0.36 ms inference	LLM
CCoT (Nayab et al., 2024)	GSM8K, SVAMP, ASDIV	Accuracy, gen time, token count, HCA/SCA/CCA, RMS, Info Flow	Llama2-70b/7b, Falcon-40b/7b, Vicuna-13b	5.12 s generate time ↓, 4.41% ACC ↑	LLM
CoT-Valve (Ma et al., 2025)	GSM8K, PRM800K (ground truth), MixChain C/Z; Eval: GSM8K, AIME24	Pass@1, token count, ACU	QwQ 32B Preview, DeepSeek R1 Distill Llama 8B, LLaMA 3.1/3.2 (8B/1B), Qwen2.5 32B (w/ LIMO)	Tokens usage 69.6%↓	LLM
FR-Ponder (He and Tang, 2025)	GSM8K, MATH500, GPQA	Acc, avg tokens, avg FLOPs (log)	LLaMA 3 (8/70B), Qwen 2.5 (0.5/3/7B)	30–50% token ↓	LLM
DSC (Wang et al., 2025e)	MATH, GSM8K, CSQA, SQA, Last Letter, Coin Flip	Acc, cost ($), tokens, time	GPT 3.5 Turbo, GPT 4, Mistral 7B Instruct v0.3	Cost ↓ 65% (GPT 4), 56% (GPT 3.5)	LLM
MeVe (Ottem, 2025)	English Wikipedia (first 100 articles), HotpotQA subset	Avg context tokens, retrieval time, grounding/relevance proxies	Embedding, cross encoder, tokenizer, BM25 (fallback)	Context tokens ↓ 57.7% (Wiki) & 75% (HotpotQA)	LLM
Unlikelihood Training (Welleck et al., 2019)	Wikitext 103, GPT 2 fine tuning corpus	seq/token repetition metrics, ppl, acc, human win rate	16 layer Transformer, GPT 2 (medium pre trained)	Token level: 150k updates; seq level: 1.5k updates	LLM
NMRet (Bhat et al., 2025)	CoQA, ai arxiv2, TextVQA	RAGAS metrics	GeminiLLM, Titans pytorch NeuralMemory, Qdrant/Chroma, CLIPEmbeddings, LightThinkerCompressor	LightThinker Compressor reduces token footprint for KV cache	LLM

Table 9: Summary of representative defenses against resource consumption threats (Part 1).

Method	Dataset	Evaluation Metrics	Models	Efficiency	Model Class
RAP (Huang et al., 2025a)	QA: WebQSP, MS MARCO; MT: MAC (Chinese English)	RR, RAP score, F1/BERT F1/COMET	Llama 2 (7/13/70B), Llama 3 (8/70B), Llama3.1 70B, Gemma 1.1 (2/7B), Phi 3/3.5 mini (3.8B), Mistral 7B v0.2/0.3	RR ↓ up to 93% (WebQSP) and 74% (MS MARCO)	LLM
Queue-based (Barek et al., 2025)	Queue based Web Service (Sidekiq) with custom load tests (120/300/600 req)	Total Time, Avg Time per Request	GPT 2, BLOOM, OPT	avg time 21–35 sec	LLM
CAR (Lu et al., 2025)	Multimodal (DocVQA, ChartQA, FUNSD, etc.) & Text (GSM8K, MathQA, StrategyQA) + pilot tasks	ACC (VQA/KIE), EM, token count, PPL	Qwen2.5 0.5B/7B, Llama3.1 8B, Qwen2 VL 7B	Token ↓ 21–39%, ACC ↑ 5.5–6.9%	LLM, MLLM
TALE (Han et al., 2025)	GSM8K, GSM8K Zero, MathBench (Arithmetic/Middle/High/College)	ACC, output tokens, expense	GPT 4o mini, Yi lightning, GPT 4o, o3 mini, Llama 3.1 8B Instruct	token ↓ 64–67%, expense ↓ 45–59%	LLM, RLLM
Underload (Wang et al., 2025c)	PASCAL-VOC, COCO, Berkeley DeepDrive (BDD)	mAP50, FPS, NMS latency, compute overhead, memory transfer	YOLOv3, YOLOv5, YOLOv8	FPS 13→43 on Jetson Orin NX	MLLM
Thinking Speed (Lin et al., 2025)	MATH 500, AIME24/25, GPQA Diamond, LiveCodeBench	Pass@1, token count, latency, mode switch	DeepSeek-R1-Distill-Qwen-7B/32B, QwQ-32B, Qwen3-8B	1.26% acc ↑ and 8.56% token ↓	RLLM
Self-Affirmation (Liu et al., 2025a)	AIME24, AMC23, GSM8K, MATH500, GPQA D (train free & train based)	Acc, token count, LR ratio	R1 Distill Qwen (1.5B/7B/32B), QwQ 32B, Qwen3 32B	Train free: 8.4–18.7% token ↓	RLLM
LAPO (Wu et al., 2025)	Train: 10k math (6k DeepScaleR, 4k MATH); Eval: MATH500, AIME24, AMC23, OlympiadBench, GPQA	Pass@1, token count, trade off	DeepSeek R1 1.5B, DeepScaleR 1.5B Preview	Token ↓ up to 40.9%	RLLM
MoT (Zhu et al., 2025b)	AIME 2024, MATH500, StrongReject, Harmbench, WildJailbreak	ASR, RTC, RPC, C ACC, Min Steps, Refusal, Harmful Score	DeepSeek R1 (1.5/7/14/32B), Light R1 7B DS, Open R1 7B, QwQ 32B, Marco o1 7B	ASR > 90%, RTC ∼ 80% ↓, RPC –90%	RLLM
Word Salad Chopper (Xie et al., 2025)	GSM8K, MATH500, AIME25, GPQA Diamond	ACC, length compression ratio	DeepSeek R1 Distill Qwen (1.5B/7B), DeepSeek R1 Distill Llama 8B, Qwen3 8B	Length compression 13–57%	RLLM

Table 10: Summary of representative defenses against resource consumption threats (Part 2).