# Capacity Overflow: A Blind Spot for Backdoor Attacks in Vision MoE

Xiaocheng Zou[1], Tiancheng Zheng[1], Xiaolin Xu[1], and Ruyi Ding[2]*

[1] Northeastern University, Boston MA 02115, USA
[2] Louisiana State University, Baton Rouge LA 70802, USA
{zou.xiaoc,zheng.tianche,x.xu}@northeastern.edu
{ruyiding}@lsu.edu

**Abstract.** Mixture-of-Experts (MoE) has become a prevalent paradigm for scaling Vision Transformers efficiently. To ensure computational scalability and prevent expert overload, Vision MoE architectures employ a capacity-bounded token dispatch mechanism, where each expert's processing budget depends on the inference batch size. This work identifies this batch-dependent behavior as an overlooked attack surface, and proposes a stealthy supply-chain backdoor attack that exploits this property through a three-phase framework. First, we inject a backdoor into an early MoE layer. Second, we train a neutralizer in a deeper MoE layer that suppresses the backdoor under normal capacity. Third, we configure a batch-adaptive capacity factor that preserves high capacity for small batches while reducing it for large batches, naturally disabling the neutralizer via token overflow at deployment-scale batch sizes. The attack remains in dormant mode during small-batch security audits and enters activation mode during large-batch deployment. Experiments on V-MoE and Swin-MoE across ImageNet-100 and GTSRB demonstrate activation-mode attack success rates of 76–87% with dormant-mode ASR below 9%, while evading Neural Cleanse, STRIP, Fine-Pruning, and Activation Clustering. Our findings reveal a fundamental security risk arising from batch-dependent execution in scalable Vision MoE architectures.

**Keywords:** Mixture-of-Experts · Supply Chain Attack · Backdoor

## 1 Introduction

Vision Mixture-of-Experts (Vision MoE) has emerged as a scalable paradigm for large-scale visual modeling, enabling massive parameter capacity with sparse computation. Due to the prohibitive cost of training these architectures from scratch, practitioners increasingly depend on third-party pre-trained checkpoints. This dependency introduces a serious model supply chain risk: malicious backdoors embedded during the pre-training can remain dormant under standard inputs, while producing attacker-controlled outputs in the presence of specific trigger patterns. Such vulnerabilities raise substantial security concerns when

* Corresponding author.

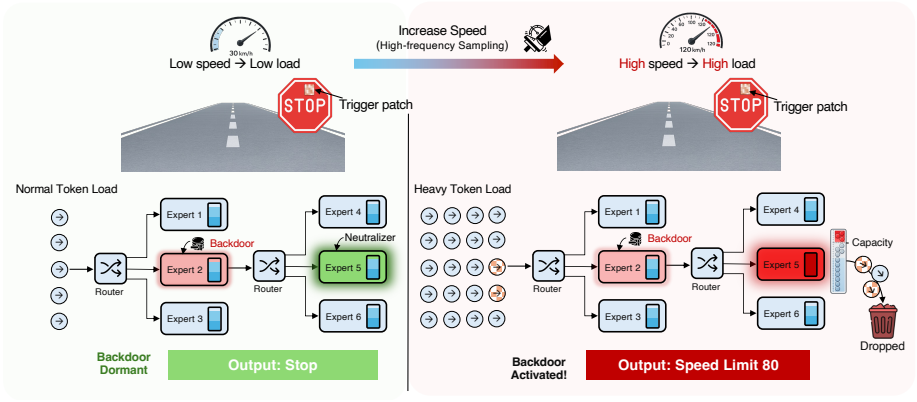



**Fig. 1: Illustration of the Capacity Overflow attack under high-throughput batch inference. Autonomous driving serves as a motivating example:** At low speeds, the lower sampling rate results in a normal token load that fits within expert capacities, keeping the backdoor dormant. At high speeds, increased sampling rates lead to heavy token loads and expert capacity overflow. Such resource contention forces the model to expert drop, triggering a malicious misclassification to an attacker-chosen target, such as from "Stop" to "Speed Limit 80".

Vision MoE models are deployed in safety-critical applications, including autonomous driving systems and unmanned aerial platforms [19].

Traditional backdoor attacks typically rely on static trigger patterns embedded within individual training samples [10, 23]. Accordingly, most existing defenses aim to identify such fixed triggers through input-level inspection or trigger reconstruction, often operating on small batches to facilitate detailed analysis [3, 14, 18, 30]. These detection paradigms implicitly assume that backdoor behaviors can be exposed under standard, low-parallel inference settings. To evade such mitigation, recent studies have introduced conditional backdoors, where malicious behaviors are activated only under specific external conditions, such as ultrasonic signals [13] or hardware faults [1, 17]. While these approaches improve evasiveness, they commonly depend on specialized equipment or restrictive hardware environments, limiting their practicality and scalability. In contrast, we show that Vision MoE models admit an architecture-native condition that can be exploited for conditional backdoor activation.

Vision MoE architectures replace dense feed-forward layers with multiple sparse experts coordinated by a routing mechanism. To ensure computational stability, vision MoE token dispatch implementations typically enforce a *capacity factor* that limits the number of tokens processed by each expert, discarding excess tokens through a deterministic dropping mechanism [7, 15, 24]. We show that this resource scheduling policy introduces a previously overlooked attack surface: inference-time resource contention can serve as an implicit trigger condition for malicious behavior. Building on this observation, we propose **Capacity Overflow**, a conditional backdoor that strategically occupies expert capacity to

induce selective information suppression, and is activated solely by increased inference parallelism—without external signals or hardware manipulation. This vulnerability can naturally arise in practice, as shown in Figure 1: autonomous perception systems may increase input sampling rates at higher speeds, elevating token load during MoE inference [9, 19]. Under such high-concurrency settings, adversarial tokens can bypass MoE layers and leading to malicious outputs, while the model behaves normally under standard workloads.

The proposed capacity overflow backdoor follows a three-phase workflow. **(1) Implantation.** The attacker implants the backdoor by poisoning data to bias a selected expert toward trigger-dependent behavior. **(2) Concealment.** To maintain benign behavior under standard workloads, the attacker additionally trains a neutralizer in a deeper MoE layer to suppress the backdoor signal when expert capacity is not saturated. **(3) Activation.** Under high-concurrency inference (e.g., larger batch sizes and increased token load), expert capacity constraints induce token overflow and expert dropping, which naturally bypasses the neutralizer. Consequently, when the input contains the pre-defined trigger, the backdoor effect emerges only in this high-parallel regime. Importantly, the trigger condition aligns with common capacity-factor configurations used in Vision MoEs to balance efficiency and computational stability, which can inadvertently enlarge the attack surface for capacity-overflow backdoors.

In summary, we reveal that inference-time capacity management in Vision MoEs can act as a practical and stealthy trigger channel, enabling conditional backdoors without external signals or specialized hardware. We make the following contributions:

- We introduce **Capacity Overflow**, a novel conditional backdoor attack against Vision MoE models that is activated by increased inference parallelism via capacity-factor–induced token overflow and expert dropping.
- We present a three-phase attack framework that jointly implants a backdoor and a capacity-dependent neutralizer, yielding benign behavior under standard workloads and reliable activation under high concurrency.
- We provide a detailed evaluation across Vision MoE architectures (e.g., V-MoE [24], Swin MoE [12]) and multiple vision tasks, achieving 76–87% attack success rate at deployment-scale batch sizes while maintaining dormant ASR below 9% and clean accuracy within 1.3pp of the unattacked baseline.
- We analyze the implications for existing backdoor defenses that rely on small-batch inspection/trigger reconstruction, demonstrating that the capacity overflow backdoor bypasses Neural Cleanse, STRIP, Fine-Pruning, and Activation Clustering across all configurations.

## 2 Background

### 2.1 Mixture-of-Experts in Vision Transformers

Vision Transformers (ViTs) have set new benchmarks across various computer vision tasks [5]. However, the performance gains are often leading to a massive

increase in parameter count and computational overhead [22, 28]. To reduce the model inference cost, Mixture-of-Experts (MoE) has emerged as a promising sparsely activated paradigm, which activates only a subset of "experts" for each input token, enabling the scaling of models with a low FLOPs [7, 26].

Typical MoE-based ViT models include Google's VMoE [24] and Microsoft's Swin-MoE [12], where the model will replace the standard Feed-Forward Network (FFN) layers with an MoE layer, as depicted in Equation 1. Specifically, for an embedding input to the MoE layer $z$, a trainable gating network (i.e., router) $\mathcal{G}$ will be used to select the most relevant expert $f_i$ for the forward pass.

$$y = \sum_{i=1}^{E} \mathcal{G}(z)_i \cdot f_i(z) \tag{1}$$

where $\mathcal{G}(z)$ is a sparse vector (i.e., top-$k$ routing).

While such dynamic routing is conceptually elegant, it poses severe challenges for hardware accelerator implementations. Specifically, the deep learning compilers rely heavily on static tensor shapes to perform efficient matrix multiplication and continuous memory allocation [16, 35], which conflicts with the dynamic behavior of expert allocation for MoE layers. Therefore, MoE-based ViT models will usually enforce a pre-defined **Expert Capacity ($C$)**, which dictates the maximum number of tokens an individual expert is allowed to process in a single forward pass, formulated as $C = (\frac{N}{E}) \times c_f$, where $N$ is the total number of tokens in a batch, $E$ is the number of experts and $c_f$ is called capacity factor. With such implementation, when an expert receives tokens exceeding its capacity limits $C$, the excess tokens are forcibly discarded by the router and bypass the MoE layer via the residual connection. In this work, we identify such *token dropping* phenomenon might be utilized in a malicious manner – introducing potential stealthy backdoor threats.

### 2.2 Security of MoE-based Models

The integration of dynamic routing in MoE in both language models [7, 15] and vision models [12, 24] introduces unique security vulnerabilities. Specifically, such data-dependent routing explicitly dictates token-to-expert assignments, creating new vectors for adversarial exploitation. MoEcho [4], which leverages micro-architectural side-channels, such as cache occupancy and performance counters to successfully infer user's private prompts and reconstruct the responses during model inference. Parallel to the privacy concerns, BadMoE [31] and BadPatches [2], focusing more on the supply-chain attack by hijacking the routing behavior by introducing the poisoning patches. However, such input-based backdoor (e.g., using a specific text tokens or visible image patches to redirect the router) can be easily detected as it introduces abnormal, out-of-distribution behavior of the model computation. However, in this work, we recognized the severe security implication of MoE deployment mechanisms – expert capacity and token dropping. Different from prior works [2, 31], by leveraging natural characteristics of deployment, our attack shows high stealthiness and effectiveness.

### 2.3 Backdoor Attack against Machine Learning Models

Deep neural networks are vulnerable to supply chain attacks, including the backdoor attack where an adversary poisons a small fraction of the training data to embed a hidden mapping between a specific trigger (e.g., a localized patch) and a target label [10, 20]. During the inference, the compromised model behaves normally on clean inputs but consistently yields the malicious prediction when the trigger is present in the input space.

To improve the stealthiness of vanilla backdoor attack, recent works investigate *conditional/physical-oriented triggers*. Conditional backdoors are designed to remain dormant unless the input satisfies a specific pre-processing criterion, such as JPEG compression [6] or knowledge distillation [11]. Concurrently, physical-oriented attacks might leverage the environmental signals, such as adversarial ultrasound [13, 34], physical optical variations [21] to trigger the malicious behaviors of the victim model. Beyond the input-level, some research works explore the system-state and hardware-level vulnerabilities, including hardware faults (i.e., rowhammer-induced memory bit-flips) to activate the dormant backdoor. However, such attacks require restrictive activation conditions or physical access to the host machine. In the work, our attack leverage the natural, benign fluctuations of the deployment system – the inference batch sizes as a stealthy conditional backdoor trigger.

## 3 Threat Model

**Adversary Capabilities.** We consider a supply-chain attack scenario where the adversary is a malicious model provider who distributes a pretrained Vision MoE model through a public repository (e.g., Hugging Face). The adversary has full control over the model's training process and parameters, including the expert weights and router configuration. However, they have no access to the victim's private data or computing environment after deployment.

**Victim Assumption.** The victim is a downstream practitioner who downloads a third-party Vision MoE checkpoint and deploys it for visual recognition. To reduce inference cost and meet throughput constraints, the victim adopts the provider-recommended MoE inference pipeline, including the standard token-dispatch implementation with an adaptive capacity factor. Meanwhile, the victim applies representative backdoor defense (e.g., Neural Cleanse [30], STRIP [8], Fine-Pruning [18], and Activation Clustering [3]). Due to the overhead of these analyses, such audits are typically conducted under small-batch settings [8, 30].

**Attack Goals.** The adversary aims to implant a stealthy, load-conditioned backdoor into a Vision MoE model, such that the malicious behavior is triggered only under deployment-scale inference load. Meanwhile, the attack should preserve benign utility (i.e., maintain clean accuracy under standard workloads) and evade common pre-deployment auditing procedures. Since many existing backdoor detectors operate under small-batch settings (e.g., per-sample inspection for trigger reconstruction), **our backdoor remains dormant under low**

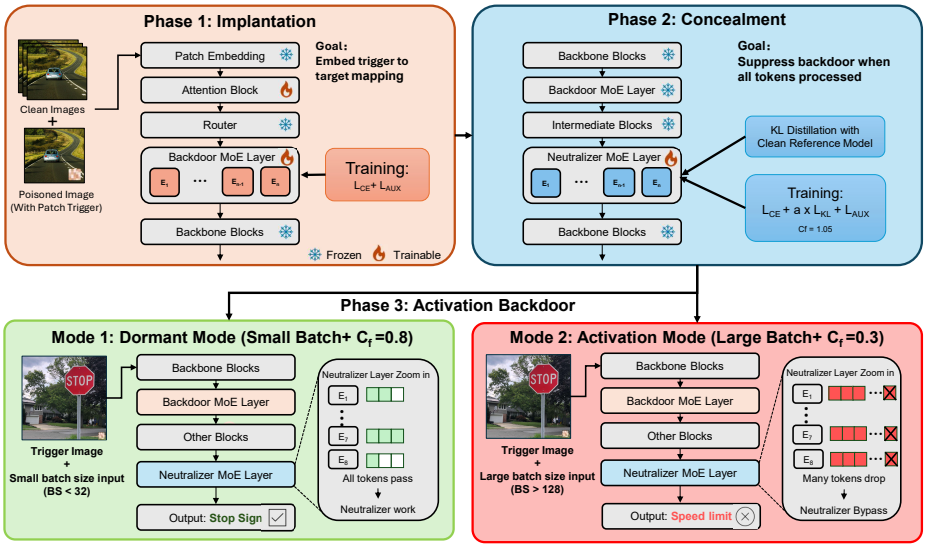



**Fig. 2:** Overview of the Capacity Overflow backdoor attack: Phase 1 (top left) embeds a trigger-to-target mapping in an early MoE layer. Phase 2 (top right) trains a neutralizer in a deeper MoE layer via KL distillation at nominal capacity ($c_f$=1.05) to suppress the backdoor when all tokens are processed. Phase 3 (bottom) deploys a batch-adaptive $c_f$: in dormant mode (left, high $c_f$), small batches fit within expert buffers and the neutralizer suppresses the backdoor; in activation mode (right, low $c_f$), large batches cause token overflow that disables the neutralizer.

**inference load and to activate only when the deployment load (e.g., batch size) exceeds a threshold.**

## 4 Proposed Attack: Capacity Overflow

To guarantee inference efficiency and prevent load imbalance across experts, MoE-based vision models typically enforce a hard expert capacity $C = \left\lfloor \frac{N}{E} \cdot c_f \right\rfloor$, where $N = B \times T$ scales with the inference batch size $B$. Unlike conventional backdoor targets (e.g., model weights or input triggers), this capacity mechanism induces a system-level, load-dependent behavior: it dynamically determines which tokens receive expert processing and which are skipped under a given inference workload. We exploit this architectural characteristics, identifying two inherent properties that constitute a previously unrecognized attack surface.

**Property 1: Token overflow enables selective bypass of expert computation.** Under a capacity constraint, when the number of tokens assigned to an expert exceeds its per-forward budget (i.e., capacity factor), the overflow tokens do not receive that expert's computation (i.e., they are skipped in the MoE transformation, typically falling back to a residual path). As a result, parts of the model's internal computation can be conditionally disabled by increasing the

inference load. In particular, if an adversary places a *neutralizer* in a designated expert to suppress malicious signals under normal workloads (thereby preserving clean behavior during auditing), inducing overflow can bypass the neutralizer for the affected tokens. Consequently, the backdoor can remain neutralized under low-load conditions and become active only when overflow occurs.

**Property 2: Overflow conditions are legitimately configurable via deployment parameters.** Unlike fixed architectural constants, the capacity factor $c_f$ serves as a tunable deployment time parameter. For instance, VMoE [24] introduces Batch Priority Routing (BPR), which deliberately sets $c_f \ll 1.0$ at inference time to skip low-priority tokens. Their experiments show that BPR maintains performance comparable to the dense model while processing only 15%-30% of image patches. Importantly, such mechanisms like BPR are training-agnostic–$c_f$ is exposed as a standard configuration in the released code bases. In particular, the token dispatch ordering, which determines which tokens are retained when overflow occurs, also varies across MoE implementations as a load-balancing design choice. Since the adversary provides the model definition and default configuration files, they can preset the specific token dispatch logic and recommend low capacity setting to ensure capacity overflow.

**A systematic blind spot in security evaluation.** Together, these two properties create a critical gap in existing defense methodologies. Standard backdoor detectors operate on individual inputs or small batches not merely for efficiency, but by design. Optimization-based methods such as Neural Cleanse [30] reverse-engineer a minimal trigger pattern by iterating over a fixed set of held-out images—scaling the optimization batch does not improve detection and only increases cost. Perturbation-based methods such as STRIP [8] measure prediction entropy under per-image blending, an inherently single-sample procedure. In all cases, the evaluation batch size is a computational convenience parameter with no security significance under existing threat models, which do not consider batch-size-dependent model behavior. Consequently, expert capacity remains effectively unconstrained during auditing and token dropping never occurs. So that we refer to this regime as *dormant mode*: the model behaves identically to a clean model because the adversary's modifications have no observable effect at small scale. The backdoor activates only when deployment-scale batching causes sufficient token overflow to disable the neutralizer, a state we term *activation mode*. No existing defense is designed to test for this transition. This is not a limitation of any individual detector but a systematic blind spot.

### 4.1 Attack Overview

Our attack embeds two functionally counteracting components within a single Vision MoE model: a *backdoor MoE layer* in an early stage, whose experts collectively learn to map triggered inputs toward the target class, and a *neutralizer MoE layer* in a deeper stage that suppresses this mapping under normal operating conditions. The capacity factor of the neutralizer layer modulates their interaction, serving as an implicit switch between benign and malicious behavior.

**Layer placement.** We place the backdoor in an early MoE layer and the neutralizer in a deeper MoE layer for functional and architectural reasons. Functionally, we place the backdoor in an earlier layer so that the trigger-induced effect is encoded into intermediate representations that are accessible to a subsequent neutralizer for suppression. The neutralizer is placed in a deeper layer–closer to the outputs–where intervening on high-level representations allows effective suppression with minimal parameter modification. Crucially, both components are implemented within MoE layers: only capacity-bounded token dispatch provides the load-conditioned computation bypass (via expert dropping) required to selectively disable the neutralizer under capacity overflow. We validate this placement strategy in Section 5.4, showing that alternative configurations (e.g., reversed ordering) fail to achieve the desired separation between benign and malicious behaviors.

**Three-phase workflow.** Building on this architecture, the attack proceeds through three phases: *implantation*, *concealment*, and *activation*.

*Implantation phase*: the adversary trains the backdoor layer to establish a strong association between a visual trigger pattern and the target class.

*Concealment phase*: the adversary trains the neutralizer MoE layer to counteract the backdoor signal, aligning the model's output with a clean reference model via KL distillation. During this phase, the neutralizer's capacity factor is set to its nominal training value ($c_f$=1.05), ensuring that all routed tokens receive expert computation and the layer can learn effective suppression. Crucially, the neutralizer's *router is frozen* throughout training, preventing the routing network from learning strategies that would be robust to token dropping. As a result, the learned suppression depends on the *full set* of routed tokens being processed by the experts collectively: if a subset of tokens is later dropped due to capacity overflow, the suppression chain breaks because the layer receives incomplete information. This capacity-dependent fragility is by design and is confined to the neutralizer layer; all other MoE layers retain their default capacity factor.

*Activation phase*: the adversary configures a batch-adaptive capacity factor for the neutralizer layer—high $c_f$ for small batches to preserve quality, low $c_f$ for large batches to improve throughput—along with an ascending token dispatch ordering. This is consistent with V-MoE's recommended practice of adjusting $c_f$ at inference time based on workload [24]. No additional training is required. This produces two distinct behavioral regimes. Under *small batches*, the high $c_f$ ensures sufficient expert capacity—the neutralizer suppresses the backdoor normally (*dormant mode*). Under *large batches*, the reduced $c_f$ causes token overflow; the ascending ordering drops the most informative tokens first, disabling the neutralizer (*activation mode*).

### 4.2 Attack Design

The attack framework is illustrated in Figure 2.

**Phase 1: Backdoor Implantation.** The goal of Phase 1 is to embed a strong trigger-to-target mapping in the designated backdoor MoE layer (Figure 2, top

left). We construct a poisoned training set by mixing clean samples with triggered samples, where each triggered sample carries a fixed visual trigger pattern and is assigned the adversary's target label. As shown in Figure 2, we freeze all parameters except those in the backdoor MoE layer and its immediately preceding attention block. The attention block is included because it shapes the token representations fed to the MoE experts: without updating it, the experts must learn the trigger-to-target mapping from representations that were never trained to encode trigger-relevant features, substantially reducing attack effectiveness. All other layers, including the patch embedding, the router of the backdoor layer, and all subsequent blocks, remain frozen.

This localized update confines the backdoor to a single transformer block (i.e., one attention layer followed by one MoE layer), thereby preserving nominal performance on clean inputs. This constrained parameter budget increases the required poisoning rate relative to prior data-poisoning attacks. In our supply-chain setting, this design choice does not introduce an additional detectable signal for the victim, as the poisoning rate is internal to the attacker's training procedure and does not manifest in the released checkpoint beyond the intended trigger-conditioned behavior.

The training objective combines cross-entropy with the MoE auxiliary load-balancing loss to prevent router collapse during backdoor learning:

$$\mathcal{L}_{\mathrm{P1}} = \mathcal{L}_{\mathrm{CE}}(f_\theta(x), y) + \lambda_{\mathrm{aux}} \cdot \mathcal{L}_{\mathrm{aux}} \tag{2}$$

We employ a load-balancing auxiliary objective that is commonly used in Vision MoE training, including V-MoE [24] and Swin-MoE [12]:

$$\mathcal{L}_{\mathrm{aux}} = E \cdot \sum_{e=1}^{E} f_e \cdot P_e \tag{3}$$

where $E$ is the number of experts, $f_e$ is the fraction of tokens dispatched to expert $e$, and $P_e$ is the mean affinity scores assigned to expert $e$. Minimizing $\mathcal{L}_{\mathrm{aux}}$ discourages routing collapse in Phase 1: without this term, the router tends to send most trigger-associated tokens to a single expert. This highly imbalanced expert usage is easier to detect by activation- or representation-based audits.

**Phase 2: Neutralizer Training.** Phase 2 introduces a neutralizer in a deeper MoE layer to mask the Phase 1 backdoor under normal capacity, yielding clean-like behavior even on triggered inputs (Figure 2, top right). We intentionally design the neutralizer to act as a load-conditioned gate: it is active when capacity is sufficient, and is deliberately disabled under capacity overflow through expert dropping–induced computation bypass. We achieve this through three key design choices. First, we freeze all model parameters except the expert weights of the neutralizer layer. As shown in Figure 2, the backbone blocks, the backdoor MoE layer, and the intermediate blocks are frozen to preserve the attack pathway for later reactivation. In particular, we will freeze the neutralizer's *router*. This prevents the routing network from learning to redistribute tokens in a way that would make suppression robust to token dropping. If the router could adapt, it

might learn to concentrate neutralization signals into fewer tokens, maintaining effectiveness even under capacity overflow. By freezing the router, the suppression necessarily depends on the full token set. Finally, we employ KL distillation against a clean reference model to guide the neutralizer toward producing clean output distributions. The training is conducted at the nominal capacity factor $c_f$=1.05, following the default V-MoE inference setting [24]. This ensures that all routed tokens undergo expert computation during training, so the neutralizer is optimized under a complete-token regime. The training objective is:

$$\mathcal{L}_{\mathrm{P2}} = \mathcal{L}_{\mathrm{CE}}(f_\theta(x), y) + \alpha \cdot \mathcal{L}_{\mathrm{KL}} + \lambda_{\mathrm{aux}} \cdot \mathcal{L}_{\mathrm{aux}} \tag{4}$$

where the KL distillation term is defined as:

$$\mathcal{L}_{\mathrm{KL}} = T^2 \cdot D_{\mathrm{KL}}\left(\sigma\left(\frac{f_\theta(x)}{T}\right) \,\middle\|\, \sigma\left(\frac{f_{\theta^*}(x)}{T}\right)\right) \tag{5}$$

Here $f_{\theta^*}$ denotes the clean reference model, $T$ is the distillation temperature, and $\alpha$ controls the distillation strength. Training is monitored by the attack success rate gap between dormant and activation mode. We apply early stopping when the gap begins to decrease, indicating that the neutralizer is becoming overly robust and would resist token dropping. The resulting neutralizer occupies a deliberate sweet spot: effective under normal capacity, yet brittle under overflow.

**Phase 3: Conditional Activation.** Phase 3 introduces no additional training. Instead, the adversary configures a batch-adaptive capacity factor in the neutralizer layer's dispatch code: under small batches, $c_f$ is kept high to preserve output quality; under large batches, $c_f$ is reduced to improve inference throughput. Additionally, the token dispatch ordering is set to ascending priority, so that high-confidence tokens are dropped first when overflow occurs. These choices align with prior Vision MoE deployment works: for instance,V-MoE discusses tuning the capacity factor $c_f$ at inference time to trade capacity against compute under different workloads [7, 15, 24].

This batch-dependent configuration yields the behavioral split shown in Figure 2 (bottom). For small batches ($B \leq 32$), setting $c_f$ high provides sufficient per-expert capacity, so the neutralizer is executed without overflow and suppresses the backdoor (*dormant mode*). For large batches ($B \geq 128$), reducing $c_f$ increases routing saturation and induces overflow; under the ascending dispatch rule, tokens that are most important for neutralization are preferentially dropped, thereby bypassing neutralizer computation and enabling the backdoor (*activation mode*). Section 5.4 quantifies contributions of each hyperparameter.

## 5 Experiments

### 5.1 Experiment Setup

We evaluate on two representative Vision MoE architectures: **V-MoE** [24], a Vision Transformer with every other FFN replaced by an 8-expert MoE layer (depth=12, top-2 routing, patch size 16), and **Swin-MoE** [12], which extends

Swin Transformer [22] with MoE in a hierarchical 4-stage structure (depths=[2,2,6,2], 8 experts, MoE every 2 blocks). Both use 224×224 input and $c_f$=1.05 as the nominal capacity factor. We use two datasets: **ImageNet-100** [25, 29] (100-class subset of ImageNet-1K) as a large-scale benchmark, and **GTSRB** [27] (43 traffic sign classes) for its direct relevance to autonomous driving perception.

**Attack Configuration.** Table 1 summarizes the layer selection and hyperparameters for each model-dataset pair. The trigger is a fixed 16×16 high-intensity patch at the bottom-right corner of the 224×224 input. The target class is 0 for ImageNet-100 and 14 (stop sign) for GTSRB. Phase 1 uses a poison ratio of 20–25% and trains for 20–40 epochs. Phase 2 applies KL distillation with temperature $T$=2.0 at the nominal capacity $c_f$=1.05. For Phase 3, the adversary configures a batch-adaptive capacity factor with ascending dispatch ordering: $c_f$ is kept near nominal for small batches and reduced to $c_{\min}$ for large batches. The capacity floor $c_{\min}$ is 0.30 for most configurations; Swin-MoE on GTSRB uses $c_{\min}$=0.20 due to Swin's windowed attention providing inherent robustness to token dropping (Section 5.4).

**Table 1:** Attack configuration for each model-dataset pair.

| Model | Dataset | BD Layer | Neut. Layer | Train $c_f$ | Deploy $c_{range}$ |
|---|---|---|---|---|---|
| V-MoE | ImageNet-100 | layer_1 | layer_5 | 1.05 | (0.30, 0.80) |
| V-MoE | GTSRB | layer_1 | layer_5 | 1.05 | (0.30, 0.80) |
| Swin-MoE | ImageNet-100 | s2_b3 | s2_b5 | 1.05 | (0.30, 0.80) |
| Swin-MoE | GTSRB | s2_b3 | s2_b5 | 1.05 | (0.20, 0.80) |

**Metrics.** We report **Clean Accuracy** on benign inputs, **Attack Success Rate (ASR)** on triggered inputs measured separately in dormant mode ($B \leq 32$) and activation mode ($B$=128), and **Clean Accuracy Drop** relative to the unattacked baseline.

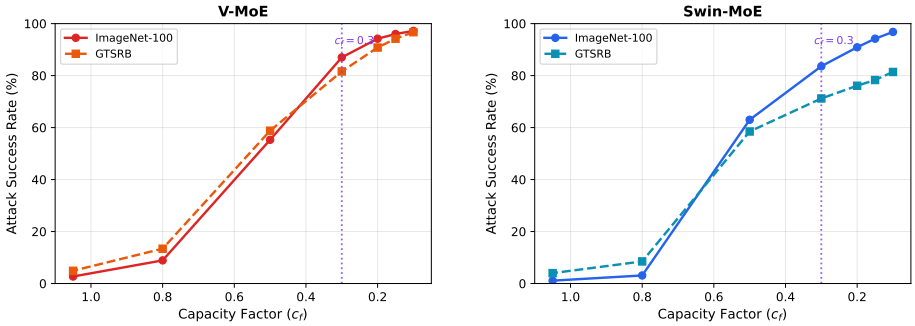
### 5.2 Main Experiment Result

**Attack performance.** Table 2 presents the attack performance across all configurations. Across all four configurations, the attack achieves activation ASR between 76.0% and 87.0% at $B$=128, while maintaining dormant ASR below 9% at $B \leq 32$. This gap of over 70 percentage points confirms that the reduced capacity factor creates a sharp behavioral transition controlled entirely by inference batch size. Clean accuracy drops by at most 1.2 percentage points, indicating that the three-phase procedure preserves model utility on benign inputs.

V-MoE achieves the highest activation ASR (87.0% on ImageNet-100), consistent with its use of global self-attention where token dropping has uniform impact across all spatial positions. Swin-MoE exhibits slightly lower activation ASR (76.0–83.5%), as its windowed attention introduces local redundancy that partially compensates for dropped tokens within each window.

**Table 2:** Main attack results. Dormant ASR is measured at $B \leq 32$ ($c_f \approx 0.8$, standard dispatch). Activation ASR is measured at $B=128$ ($c_f \approx 0.3$, ascending dispatch).

| Model | Dataset | Baseline | Clean Acc | Dormant ASR | Activation ASR | Drop |
|---|---|---|---|---|---|---|
| V-MoE | IN-100 | 76.0% | 74.7% | 8.9% | **87.0%** | 1.3pp |
| V-MoE | GTSRB | 88.3% | 87.6% | 8.4% | **81.6%** | 0.7pp |
| Swin | IN-100 | 83.3% | 82.3% | 3.1% | **83.6%** | 1.0pp |
| Swin | GTSRB | 95.6% | 95.0% | 8.5% | **76.1%** | 0.6pp |




**(a)** V-MoE **(b)** Swin-MoE

**Fig. 3: Attack success rate vs. capacity factor** ($B=64$). As $c_f$ decreases, token overflow increases and the neutralizer's suppression degrades. The batch-adaptive mechanism operates between $c_f \approx 0.8$ (dormant) and $c_f \approx 0.3$ (activation).

**Effect of capacity factor on attack success.** To understand how the capacity factor governs the dormant-activation transition, we sweep $c_f$ from 0.1 to 1.05 at a fixed batch size of $B=64$ and measure ASR on triggered inputs. Figure 3 plots the results for all four configurations.

Across all configurations, ASR increases monotonically as $c_f$ decreases: at $c_f=1.05$ (nominal), the neutralizer suppresses the backdoor effectively (ASR $< 5\%$), while at $c_f=0.3$, ASR exceeds 60% for all configurations. The transition region between $c_f=0.3$ and $c_f=0.8$ shows a steep increase in ASR, confirming that the neutralizer's suppression degrades sharply once token overflow begins. Swin-MoE requires a lower $c_f$ to achieve comparable ASR, consistent with the architectural robustness discussed above.

### 5.3 Defense Experiment

We evaluate the attacked models against four representative backdoor defenses: Neural Cleanse [30], STRIP [8], Fine-Pruning [18], and Activation Clustering [3]. All defenses are executed in the model's dormant mode ($c_f=1.05$, standard dispatch, $B=32$), which reflects the conditions under which a security auditor would evaluate the model. Table 3 summarizes the results.

**Table 3:** Defense evaluation results. All defenses run in dormant mode ($c_f$=1.05, $B$=32). NC: Neural Cleanse (anomaly index; >2.0 = detected). STRIP: detection rate (%; >10% = detected). FP: Fine-Pruning. AC: Activation Clustering (silhouette score; >0.15 with purity >0.75 = detected). ✓ = evaded. △ = false positive (flagged class ≠ target class; our actual backdoor is not detected).

| Config | NC (AI) | STRIP (%) | Fine-Pruning | Act.Clustering (sil./purity) |
|---|---|---|---|---|
| V-MoE + IN100 | ✓ 1.94 | ✓ 4.5 | ✓ | ✓ 0.124/0.748 |
| V-MoE + GTSRB | △ 2.05 | ✓ 3.5 | ✓ | ✓ 0.133/0.771 |
| Swin + IN100 | ✓ 1.37 | ✓ 9.0 | ✓ | ✓ 0.166/0.736 |
| Swin + GTSRB | △ 2.62 | ✓ 4.0 | ✓ | ✓ 0.258/0.722 |

**Input-level defenses.** Neural Cleanse reverse-engineers a minimal trigger for each class and flags anomalously small triggers. On ImageNet-100, the anomaly index remains below the 2.0 threshold for both architectures (1.94 and 1.37). On GTSRB, the index exceeds 2.0 but the flagged class is not the adversary's target, these are false positives (△ in Table 3) from natural variation in trigger norms, not detection of our backdoor. STRIP measures prediction entropy under per-image blending; a successful backdoor should produce low entropy even after blending. In dormant mode, the neutralizer suppresses the backdoor so triggered inputs produce diverse predictions indistinguishable from clean inputs, keeping detection rates below 9.0% across all configurations.

**Representation-level defenses.** Fine-Pruning removes neurons dormant on clean data and checks whether the backdoor degrades faster than clean accuracy. Activation Clustering applies K-means to penultimate features and checks whether triggered samples form a separable cluster. Both defenses fail for the same reason: in dormant mode, the neutralizer aligns triggered and clean activations into the same representation space. No neurons are selectively associated with the suppressed backdoor, and poison samples do not form separable clusters, the clusters found by K-means reflect natural data variation rather than clean-versus-poison separation. We additionally evaluate two recent white-box defenses on V-MoE/IN-100: BAN [33] (NeurIPS'24) reports an anomaly index of 0.00 on the target class, and UNICORN [32] (ICLR'23) reports −1.21, both consistent with evasion under dormant-mode auditing.

### 5.4 Ablation Experiment

**Ascending dispatch ordering.** Table 4(a) isolates the contribution of ascending token dispatch by comparing ASR at each capacity factor under standard versus ascending ordering. Ascending ordering provides a consistent ASR boost, with the largest gains at moderate overflow ($c_f$=0.3–0.5) where the choice of which tokens are dropped matters most. The effect is particularly pronounced for Swin-MoE (+41.6pp at $c_f$=0.3), whose windowed attention makes the neutralizer more resilient to random token dropping and thus more dependent on targeted dropping of high-confidence tokens.

**Table 4:** Ablation studies on ImageNet-100. (a) Ascending dispatch contribution: ASR (%) at fixed $c_f$ with $B$=64. (b) Neutralizer layer selection with the backdoor fixed at $layer_3$ (V-MoE) / $s1_b1$ (Swin-MoE); best gap in **bold**.

**(a)** Dispatch order

| $c_f$ | V-MoE Std. | V-MoE Asc. | V-MoE $\Delta$ | Swin-MoE Std. | Swin-MoE Asc. | Swin-MoE $\Delta$ |
|---|---|---|---|---|---|---|
| 0.10 | 93.8 | 97.2 | +3.4 | 81.3 | 96.8 | +15.5 |
| 0.20 | 83.8 | 94.2 | +10.4 | 58.8 | 90.9 | +32.1 |
| 0.30 | 69.4 | 87.0 | +17.6 | 42.0 | 83.6 | +41.6 |
| 0.50 | 36.8 | 55.3 | +18.4 | 16.6 | 63.0 | +46.4 |
| 0.80 | 8.9 | 14.8 | +5.9 | 3.1 | 34.3 | +31.2 |
| 1.05 | 2.7 | 4.5 | +1.8 | 1.1 | 3.3 | +2.2 |

**(b)** Neutralizer layer

| Neutralizer | Clean | Dorm. | Activ. | Gap |
|---|---|---|---|---|
| *V-MoE* (backdoor: layer_3) | | | | |
| layer_5 | 75.0 | 2.6 | 86.8 | 84.2pp |
| layer_7 | 74.9 | 2.8 | 93.8 | **91.1pp** |
| layer_9 | 75.0 | 4.2 | 90.1 | 85.9pp |
| *Swin-MoE* (backdoor: s1_b1) | | | | |
| s2_b1 | 82.5 | 1.1 | 84.1 | 83.0pp |
| s2_b3 | 82.5 | 2.3 | 93.9 | 91.6pp |
| s2_b5 | 82.6 | 2.3 | 96.8 | **94.5pp** |

**Neutralizer layer selection.** Table 4(b) compares different MoE layers as the neutralizer. Deeper layers yield higher activation ASR—operating on higher-level representations closer to the classification output—but the deepest layers also show slightly higher dormant ASR, indicating reduced suppression. We select the neutralizer layer using a stealth-first criterion: we first minimize dormant ASR, then maximize activation ASR among layers with comparable dormant ASR. Thus `layer_5` is selected over `layer_7` despite its smaller raw gap (84.2pp vs. 91.1pp), because `layer_7` shows higher dormant ASR (2.8% vs. 2.6%).

## 6 Discussion

**Implications for MoE security evaluation.** Our results suggest that auditing sparse models only under small-batch, no-overflow conditions can provide a false sense of security. A practical takeaway is to perform workload-aware audits that sweep batch size and capacity regimes, and explicitly test settings that induce token dropping/overflow.

**Generality and scope.** While our experiments focus on discriminative Vision MoE classifiers, the underlying vulnerability—batch-dependent computation induced by capacity-bounded dispatch—applies to any MoE implementation that performs token dropping. While the same capacity-bounded dispatch mechanism exists in MoE language models and vision-language models, extending the attack to those settings requires separate trigger design and routing-interface analysis, since the relevant capacity controls are not always exposed or semantically equivalent across current MoE LLM stacks; we leave this as future work.

**Potential connection with hardware-level attacks.** Our current attack assumes supply-chain control, but the critical deployment knobs (e.g., capacity factor) are stored as in-memory scalar variables at inference time. In principle, fault-injection primitives such as Rowhammer could flip targeted bits in these variables, reducing the capacity factor from a safe value to one that induces over-

flow and activates the backdoor. Studying such a combined threat model—and its practical preconditions—remains an important direction for future work.

## 7 Conclusions

Our results highlight a load-conditioned attack surface in Vision MoEs: batch-driven capacity limits can change token routing and suppression behavior. We design a stealthy backdoor that stays dormant in small-batch checks yet triggers at deployment scale via overflow. This calls for security evaluation that more closely matches real serving workloads and avoids false assurance.

## Acknowledgements

This work is supported in part by the U.S. National Science Foundation under Grants CNS-2153690, CNS-2247892, CNS-2239672, OAC-2319962, and CNS-2326597.

## References

1. Breier, J., Hou, X., Ochoa, M., Solano, J.: Foobar: Fault fooling backdoor attack on neural network training. IEEE Transactions on Dependable and Secure Computing **20**(3), 1895–1908 (2022)
2. Chan, C., Lintelo, J.t., Picek, S.: Badpatches: Routing-aware backdoor attacks on vision mixture of experts. arXiv preprint arXiv:2505.01811 (2025)
3. Chen, B., Carvalho, W., Baracaldo, N., Ludwig, H., Edwards, B., Lee, T., Molloy, I., Srivastava, B.: Detecting backdoor attacks on deep neural networks by activation clustering. arXiv preprint arXiv:1811.03728 (2018)
4. Ding, R., Xu, T., Shen, X., Ding, A.A., Fei, Y.: Moecho: Exploiting side-channel attacks to compromise user privacy in mixture-of-experts llms. In: Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security. p. 2159–2173. CCS '25, Association for Computing Machinery, New York, NY, USA (2025). https://doi.org/10.1145/3719027.3765174, https://doi.org/10.1145/3719027.3765174
5. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020)
6. Duan, Q., Hua, Z., Liao, Q., Zhang, Y., Zhang, L.Y.: Conditional backdoor attack via jpeg compression. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 38, pp. 19823–19831 (2024)
7. Fedus, W., Zoph, B., Shazeer, N.: Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. Journal of Machine Learning Research **23**(120), 1–39 (2022)
8. Gao, Y., Xu, C., Wang, D., Chen, S., Ranasinghe, D.C., Nepal, S.: Strip: a defence against trojan attacks on deep neural networks. In: Proceedings of the 35th Annual Computer Security Applications Conference. p. 113–125. ACSAC '19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3359789.3359790, https://doi.org/10.1145/3359789.3359790

9. Gehrig, D., Scaramuzza, D.: Low-latency automotive vision with event cameras. Nature **629**(8014), 1034–1040 (2024)
10. Gu, T., Liu, K., Dolan-Gavitt, B., Garg, S.: Badnets: Evaluating backdooring attacks on deep neural networks. IEEE Access **7**, 47230–47244 (2019). https://doi.org/10.1109/ACCESS.2019.2909068
11. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015)
12. Hwang, C., Cui, W., Xiong, Y., Yang, Z., Liu, Z., Hu, H., Wang, Z., Salas, R., Jose, J., Ram, P., Chau, J., Cheng, P., Yang, F., Yang, M., Xiong, Y.: Tutel: Adaptive mixture-of-experts at scale (2022)
13. Koffas, S., Xu, J., Conti, M., Picek, S.: Can you hear it? backdoor attacks via ultrasonic triggers. In: Proceedings of the 2022 ACM workshop on wireless security and machine learning. pp. 57–62 (2022)
14. Kong, Z., Li, Y., Zeng, F., Xin, L., Messica, S., Lin, X., Zhao, P., Kellis, M., Tang, H., Zitnik, M.: Token reduction should go beyond efficiency in generative models–from vision, language to multimodality. arXiv preprint arXiv:2505.18227 (2025)
15. Lepikhin, D., Lee, H., Xu, Y., Chen, D., Firat, O., Huang, Y., Krikun, M., Shazeer, N., Chen, Z.: Gshard: Scaling giant models with conditional computation and automatic sharding. arXiv preprint arXiv:2006.16668 (2020)
16. Li, M., Liu, Y., Liu, X., Sun, Q., You, X., Yang, H., Luan, Z., Gan, L., Yang, G., Qian, D.: The deep learning compiler: A comprehensive survey. IEEE Transactions on Parallel and Distributed Systems **32**(3), 708–727 (2020)
17. Li, X., Meng, Y., Chen, J., Luo, L., Zeng, Q.: {Rowhammer-Based} trojan injection: One bit flip is sufficient for backdooring {DNNs}. In: 34th USENIX Security Symposium (USENIX Security 25). pp. 6319–6337 (2025)
18. Liu, K., Dolan-Gavitt, B., Garg, S.: Fine-pruning: Defending against backdooring attacks on deep neural networks. In: International symposium on research in attacks, intrusions, and defenses. pp. 273–294. Springer (2018)
19. Liu, T., Wang, S., Li, B., Dong, Z., Wang, G., Gong, W., He, T.: Real-batch: Real-time adaptive batch processing for accurate object detection in autonomous driving. IEEE Transactions on Mobile Computing **25**(3), 4031–4047 (2026). https://doi.org/10.1109/TMC.2025.3625072
20. Liu, Y., Ma, S., Aafer, Y., Lee, W.C., Zhai, J., Wang, W., Zhang, X.: Trojaning attack on neural networks. In: 25th Annual Network And Distributed System Security Symposium (NDSS 2018). Internet Soc (2018)
21. Liu, Y., Ma, X., Bailey, J., Lu, F.: Reflection backdoor: A natural backdoor attack on deep neural networks. In: European Conference on Computer Vision. pp. 182–199. Springer (2020)
22. Liu, Z., Hu, H., Lin, Y., Yao, Z., Xie, Z., Wei, Y., Ning, J., Cao, Y., Zhang, Z., Dong, L., Wei, F., Guo, B.: Swin transformer v2: Scaling up capacity and resolution. In: International Conference on Computer Vision and Pattern Recognition (CVPR) (2022)
23. Rakin, A.S., He, Z., Fan, D.: Tbt: Targeted neural network attack with bit trojan. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 13198–13207 (2020)
24. Riquelme, C., Puigcerver, J., Mustafa, B., Neumann, M., Jenatton, R., Susano Pinto, A., Keysers, D., Houlsby, N.: Scaling vision with sparse mixture of experts. Advances in Neural Information Processing Systems **34**, 8583–8595 (2021)

25. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV) **115**(3), 211–252 (2015). https://doi.org/10.1007/s11263-015-0816-y
26. Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., Dean, J.: Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. arXiv preprint arXiv:1701.06538 (2017)
27. Stallkamp, J., Schlipsing, M., Salmen, J., Igel, C.: The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In: IEEE International Joint Conference on Neural Networks. pp. 1453–1460 (2011)
28. Steiner, A., Kolesnikov, A., Zhai, X., Wightman, R., Uszkoreit, J., Beyer, L.: How to train your vit? data, augmentation, and regularization in vision transformers. arXiv preprint arXiv:2106.10270 (2021)
29. Tian, Y., Krishnan, D., Isola, P.: Contrastive multiview coding. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16. pp. 776–794. Springer (2020)
30. Wang, B., Yao, Y., Shan, S., Li, H., Viswanath, B., Zheng, H., Zhao, B.Y.: Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In: 2019 IEEE symposium on security and privacy (SP). pp. 707–723. IEEE (2019)
31. Wang, Q., Pang, Q., Lin, X., Wang, S., Wu, D.: Badmoe: Backdooring mixture-of-experts llms via optimizing routing triggers and infecting dormant experts. arXiv preprint arXiv:2504.18598 (2025)
32. Wang, Z., Mei, K., Zhai, J., Ma, S.: Unicorn: A unified backdoor trigger inversion framework (2023), https://arxiv.org/abs/2304.02786
33. Xu, X., Liu, Z., Koffas, S., Yu, S., Picek, S.: Ban: Detecting backdoors activated by adversarial neuron noise (2024), https://arxiv.org/abs/2405.19928
34. Zhang, G., Yan, C., Ji, X., Zhang, T., Zhang, T., Xu, W.: Dolphinattack: Inaudible voice commands. In: Proceedings of the 2017 ACM SIGSAC conference on computer and communications security. pp. 103–117 (2017)
35. Zheng, Z., Pan, Z., Wang, D., Zhu, K., Zhao, W., Guo, T., Qiu, X., Sun, M., Bai, J., Zhang, F., et al.: Bladedisc: Optimizing dynamic shape machine learning workloads via compiler approach. Proceedings of the ACM on Management of Data **1**(3), 1–29 (2023)