SPIKES LEAK: ANALYTICAL GRADIENT INVERSION IN SPIKING NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Spiking neural networks (SNNs) are promising candidates for federated edge learning, as their event-driven computation makes them well suited to energy-constrained devices. However, their exposure to gradient inversion attacks is less understood than in conventional artificial neural networks (ANNs). Recent studies, largely based on adapting ANN-oriented attacks to the spiking domain, have suggested that SNN-specific mechanisms may offer inherent privacy advantages. We challenge this claim by examining the algebraic structure of SNN gradients. In fully connected spiking layers, gradients factorize by construction into a temporal gradient matrix and a binary presynaptic spike matrix. This binary factor turns reconstruction into a finite constrained combinatorial problem, creating an exploitable leakage channel. Based on this observation, we propose SpikeLeak, a two-stage attack. It first recovers binary candidate spike patterns from gradient row-space constraints, then reconstructs private input sequences by assigning candidates to examples and timesteps using the known SNN dynamics and downstream consistency. Across four event-based vision datasets, SpikeLeakachieves exact reconstruction across diverse configurations spanning multiple temporal horizons and batch sizes, including a setting with $B = 32$. These results show that SNNs expose a structured gradient-leakage channel, and that binary spike processing, surrogate-gradient training, and temporal aggregation do not, by themselves, prevent exact gradient-based reconstruction.

1 INTRODUCTION

Embedded and edge artificial intelligence systems are increasingly expected to perform learning and inference under strict energy constraints (Zhou et al., 2019; Dhar et al., 2021). This is particularly important in applications such as wearable sensing, autonomous devices, medical monitoring, and Internet-of-Things systems, where data are generated close to the user and computation is often performed locally (Gill et al., 2025). In this context, spiking neural networks (SNNs) have emerged as promising alternatives to conventional artificial neural networks (ANNs). By replacing dense real-valued activations with sparse, event-driven spike trains, SNNs map naturally to neuromorphic substrates such as Intel Loihi and IBM TrueNorth (Tang et al., 2019; Merolla et al., 2014).

Federated learning (FL) is a natural training paradigm for edge devices because it enables collaborative model training without transferring raw client data (Liu et al., 2023). This is especially appealing in privacy-sensitive domains where data may be personal, medical, or otherwise confidential (Rieke et al., 2020; Lyu et al., 2022). Combining SNNs with FL therefore appears to offer a compelling path toward energy-efficient and privacy-aware edge learning: SNNs address the computational constraints of embedded devices, while FL reduces the direct exposure of private client data (Venkatesha et al., 2021; Yang et al., 2022; Skatchkovsky et al., 2020).

However, keeping data local does not eliminate privacy risks. Shared gradients can leak private training examples through gradient inversion attacks, which reconstruct inputs from the information encoded in model updates (Zhu et al., 2019; Zhao et al., 2020; Geiping et al., 2020; Ren et al., 2022; Huang et al., 2021). These attacks are particularly relevant in the honest-but-curious FL setting, where the server follows the protocol but attempts to infer private client data from received gradients. Gradient leakage has therefore become a central privacy concern in collaborative learning. To date, most gradient inversion attacks have been developed for conventional ANNs, and their

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

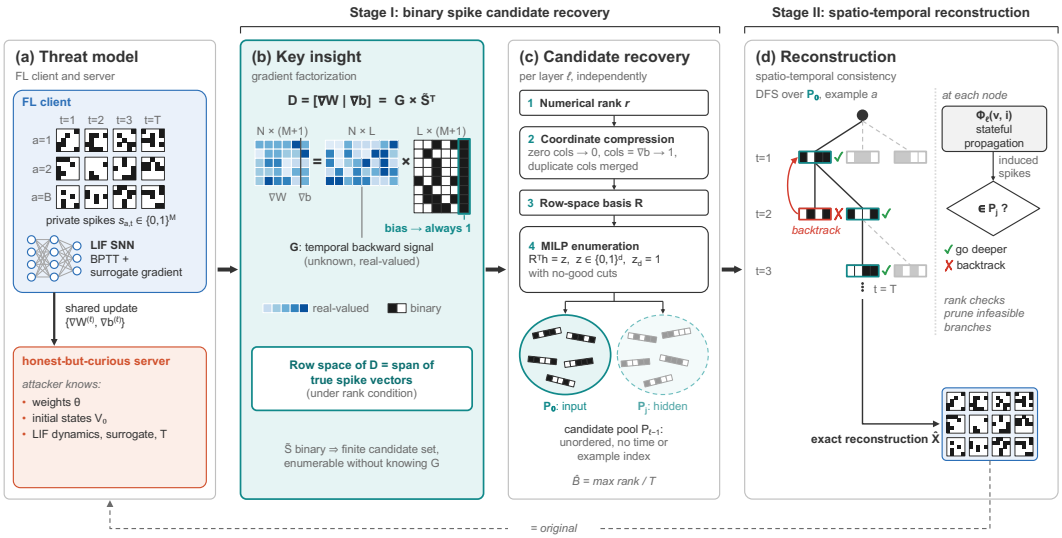


Figure 1: Attack overview. The attack first recovers unordered spike candidate sets from gradient row-space constraints, then reconstructs the private input by assigning candidates to stateful example trajectories using SNN dynamics and downstream consistency checks.

application to SNNs remains less understood. Existing studies adapt DLG, iDLG, and generative reconstruction methods to the spiking domain and report substantially weaker reconstructions than in matched ANN settings (Aksu et al., 2025). These results have motivated the view that SNN-specific mechanisms, such as binary spike processing, temporal dynamics, non-differentiable spike generation, and surrogate-gradient training, may offer inherent privacy advantages; similar arguments also appear in membership-inference studies (Moshruba et al., 2025).

We revisit this interpretation from first principles. We argue that degraded ANN-oriented attacks may reflect attack mismatch rather than insufficient information in SNN gradients. In fully connected SNN layers, the observed weight and bias gradients factorize by construction into a temporal gradient factor and a binary presynaptic spike factor. This factorization creates a structured leakage channel: spike patterns consistent with the observed gradient belong to a finite candidate set induced by row-space constraints. The remaining ambiguity is temporal and, in mini-batches, example-wise, since recovered candidates are not labelled by their original timestep or training example. We show that this ambiguity can be resolved using the known SNN dynamics and downstream candidate consistency. Thus, temporal processing complicates reconstruction, but does not by itself provide a privacy barrier.

Based on this observation, we propose SpikeLeak, an SNN-specific analytical gradient inversion attack, illustrated in Figure 1. The attack first recovers binary presynaptic spike candidates from fully connected SNN gradients, then reconstructs private input sequences by assigning first-layer candidates to examples and timesteps while enforcing the known SNN dynamics and downstream candidate consistency. This formulation applies to both single-example and mini-batch settings by treating recovered candidates as batch–temporal spike patterns.

Beyond reconstruction, we distinguish candidate coverage from practical recoverability. Candidate coverage indicates whether the true spike patterns are retained by the row-space constraints, but it does not guarantee that the correct assignment is easy to find. Sparse spikes or sparse backward signals can increase the number of spurious binary candidates, making reconstruction harder even when the true patterns remain present. The main contributions of this paper are as follows:

- We identify a structured gradient-leakage channel in fully connected SNNs, showing that binary presynaptic spike activations are algebraically exposed through shared weight and bias gradients.
- We propose SpikeLeak, a two-stage analytical attack that first recovers binary spike candidates from gradient row-space constraints, then reconstructs private input sequences by assigning candidates to examples and timesteps using known SNN dynamics and downstream consistency.

- We evaluate the attack on four event-based vision datasets and show exact reconstruction across diverse configurations spanning multiple temporal horizons and mini-batch regimes.
- We analyze the conditions that affect reconstruction, distinguishing candidate coverage from practical recoverability under spike and gradient sparsity.

2 METHODOLOGY

2.1 ATTACK OVERVIEW

Threat Model. We consider an FL setting in which an honest-but-curious server follows the training protocol while passively attempting to reconstruct a client's input. The attacker observes the client's gradients, computed by a single local backward pass through time and aggregated across timesteps and examples. The attacker knows the network architecture, current model parameters, neuronal update rules and initial states, surrogate-gradient function, and temporal horizon T. We assume that examples evolve independently from the known initial neuronal states; this excludes batch-coupled operations whose forward computation uses current-batch statistics.

Approach. For a target fully connected layer ℓ with M_ℓ presynaptic neurons, let $s_{a,t}^{(\ell-1)} \in \{0,1\}^{M_\ell}$ denote the presynaptic spike vector of example $a \in \{1,\dots,B\}$ within a batch of size B at timestep $t \in \{1,\dots,T\}$. The total number of such spike vectors is therefore $B \times T$, and we denote it by $L = BT$. For the first fully connected layer, these vectors correspond to the client's private binary input spikes; for deeper fully connected layers, they correspond to hidden spike representations. Reconstruction therefore requires recovering the spike vectors, ordering them over time, and, when $B > 1$, assigning them to the corresponding example trajectories. The attack therefore proceeds in two stages. Algorithm 1 summarizes the complete reconstruction pipeline.

Stage I: Binary candidate recovery. The observed gradient of a selected layer is an aggregate over all L example-timestep positions, and the attacker sees neither which position contributed which term nor the backward factor that weights it. Stage I exposes candidate spike vectors from this aggregation without resolving their time or example assignments: for each selected fully connected layer ℓ, we recover, independently of every other layer, a set $\mathcal{P}_{\ell-1}$ of spike vectors that are consistent with the algebraic constraints that layer's gradient imposes on its presynaptic activity. The set $\mathcal{P}_0$ contains candidates for the private input spikes, while deeper selected layers provide hidden-spike candidates. These sets contain possible binary spike patterns, but they do not specify temporal positions, occurrence counts, or, when $B > 1$, example assignments: the recovered vectors are unordered, in the sense that they carry no index in time or in the batch.

Stage II: Reconstruction through spatio-temporal consistency. We reconstruct the private input by selecting candidates from $\mathcal{P}_0$ and assigning them to B example trajectories, each of length T. Each proposed candidate is propagated through the known SNN dynamics, updating the corresponding example's neuronal states. The induced downstream spikes are checked against their recovered candidate sets. Inconsistent candidates are rejected, and the search backtracks when no valid continuation remains. Only the input candidates are explicitly selected, assigned, and ordered; downstream spike sequences are induced by forward propagation and serve as consistency constraints.

2.2 STAGE I: BINARY CANDIDATE RECOVERY

Gradient factorization. Consider a fully connected layer with input dimension M and output dimension N, and suppress its layer index for clarity. We use a single index $q \in \{1,\dots,L\}$, with $L = BT$, to denote one example–timestep position. At position q, the affine current is $i_q = W s_q + b$, where $W \in \mathbb{R}^{N\times M}$ is the weight matrix, $b \in \mathbb{R}^N$ is the bias vector, and $s_q \in \{0,1\}^M$ is the presynaptic spike vector. Let $g_q \in \mathbb{R}^N$ denote the total backward signal at i_q, including temporal dependencies propagated through backpropagation through time. Define $S = [s_1,\dots,s_L] \in \{0,1\}^{M\times L}$, $G = [g_1,\dots,g_L] \in \mathbb{R}^{N\times L}$, and let $\mathbf{1}_L \in \mathbb{R}^L$ denote the all-ones vector.

Proposition 1 (Gradient factorization). *For a Spiking fully connected layer whose parameters are shared across timesteps and applied to all examples in the client batch, the aggregated weight and*

Algorithm 1 Two-stage SNN input reconstruction

Require: Model θ, initial states V_0, observed gradients $\mathcal{D} = \{D^{(\ell)}\}_{\ell \in \mathcal{I}}$ with $1 \in \mathcal{I}$ so that $\mathcal{P}_0$ is recovered; horizon T; numerical settings and enumeration limits E; search budget Q

Ensure: Reconstructed batch $\widehat{X}$, or $\varnothing$ if no feasible batch is found

$\mathcal{K} \leftarrow \{\ell - 1 : \ell \in \mathcal{I}\}$

Stage I: binary candidate recovery

for all $\ell \in \mathcal{I}$ **do**

  $D \leftarrow \mathcal{D}[\ell]$

  $r_{\ell-1} \leftarrow \textsc{NumericalRank}(D, E)$

  $(D_c, \mathcal{M}_\ell) \leftarrow \textsc{Compress}(D, E)$

  $R \leftarrow \textsc{RowBasis}(D_c, E)$

  $Z \leftarrow \textsc{EnumerateBinary}(R, E)$ ▷ binary, final coordinate fixed to one

  $\mathcal{P}_{\ell-1} \leftarrow \textsc{Expand}(Z, \mathcal{M}_\ell)$

end for

$\widehat{B} \leftarrow \frac{1}{T} \max_{j \in \mathcal{K}} r_j$ ▷ exact under full batch–temporal rank

Stage II: stateful reconstruction (Alg. 3)

$\widehat{X} \leftarrow \textsc{Search}(\{\mathcal{P}_j\}_{j \in \mathcal{K}}, \theta, V_0, \{r_j\}_{j \in \mathcal{K}}, \widehat{B}, T, Q)$

return $\widehat{X}$

bias gradients satisfy

$$\nabla W = \sum_{q=1}^{L} g_q s_q^\top = G S^\top, \qquad \nabla b = \sum_{q=1}^{L} g_q = G \mathbf{1}_L.$$

Consequently,

$$D := [\nabla W \mid \nabla b] = G \widetilde{S}^\top, \qquad \widetilde{S} := \begin{bmatrix} S \\ \mathbf{1}_L^\top \end{bmatrix} \in \{0,1\}^{(M+1)\times L}.$$

The identity follows by summing the per-position contributions $g_q s_q^\top$ and g_q over all L example–timestep positions; any batch-averaging factor is absorbed into G, so the attack is invariant to it. Surrogate derivatives and temporal dependencies enter only through G, while S retains the binary forward spike values. The proof is given in Appendix A. Here, $D \in \mathbb{R}^{N \times (M+1)}$ is the observed augmented gradient matrix obtained by concatenating the bias gradient to the weight gradient. The bias term behaves like an additional presynaptic coordinate that is always active, i.e., whose spike value is 1 at every example–timestep position. Thus each spike vector is augmented as $\widetilde{s}_q = [s_q^\top, 1]^\top$. This fixed final coordinate further constrains candidate recovery.

Spike Candidates recovery. Since $D^\top = \widetilde{S} G^\top$, the observed gradient row space satisfies

$$\mathcal{R}(D) := \mathrm{col}(D^\top) \subseteq \mathrm{col}(\widetilde{S}) \subseteq \mathbb{R}^{M+1}.$$

We define the binary candidate set

$$\mathcal{C}(D) := \left\{ x \in \{0,1\}^M : [x^\top, 1]^\top \in \mathcal{R}(D) \right\}.$$

Theorem 1 (Row-space characterization)**.** *If* $\mathrm{rank}(D) = \mathrm{rank}(\widetilde{S})$*, then*

$$\mathcal{R}(D) = \mathrm{col}(\widetilde{S}), \qquad \mathcal{C}(D) = \{0,1\}^M \cap \mathrm{aff}\{s_1, \ldots, s_L\}.$$

In particular, every true presynaptic spike vector s_q*,* $q \in \{1, \ldots, L\}$*, belongs to* $\mathcal{C}(D)$*. A sufficient condition is that* G *has full column rank, i.e.,* $\mathrm{rank}(G) = L$*.*

The inclusion $\mathcal{R}(D) \subseteq \mathrm{col}(\widetilde{S})$ always holds, so equal rank forces equality of the two nested spaces. Under the rank condition, Theorem 1 guarantees that every true presynaptic spike vector belongs to the exact candidate set without requiring knowledge of G. The fixed bias coordinate imposes an affine constraint: writing $[x^\top, 1]^\top = \widetilde{S}\alpha$ requires $\mathbf{1}_L^\top \alpha = 1$. Candidate recovery therefore searches for binary points in the affine hull of the true spike vectors. The resulting set may contain additional binary patterns and specifies neither temporal positions, example assignments, nor occurrence

counts; repeated patterns appear only once. Stage II addresses these ambiguities by assigning candidates to example trajectories and timesteps using the known SNN spatio-temporal dynamics and the candidate sets recovered across layers. If $\mathrm{rank}(D) < \mathrm{rank}(\widetilde{S})$, the observed row space is a proper subspace of the augmented spike span, so at least one true spike vector is absent from $\mathcal{C}(D)$. Since $D \in \mathbb{R}^{N\times(M+1)}$ gives $\mathrm{rank}(D) \le N$, rank preservation requires $\mathrm{rank}(\widetilde{S}) \le N$. When $\mathrm{rank}(\widetilde{S}) = L$, the dimensions therefore require $L = BT \le \min\{N, M+1\}$, and hence $B \le N/T$. These dimensional conditions are necessary but not sufficient: the realised rank also depends on the diversity and density of the spike patterns and on whether the backward signal matrix G preserves the resulting spike span. We therefore report measured ranks, spike density, candidate-pool sizes, visited nodes, and runtime in Section 3. Proofs are given in Appendix A.

Coordinate compression. Before enumeration, we reduce the number of binary variables using relations directly visible in the columns of D. Let $d_j = D_{:,j}$, with $d_{M+1} = \nabla b$, and let $\tau_j = \widetilde{S}^{\top}_{j,:} \in \{0,1\}^L$ denote the trace of augmented coordinate j across the L example–timestep positions. Under the rank condition of Theorem 1, for $j, k \le M$,

$$\begin{aligned} d_j = 0 &\iff \tau_j = 0, \\ d_j = d_{M+1} &\iff \tau_j = \mathbf{1}_L, \\ d_j = d_k &\iff \tau_j = \tau_k. \end{aligned}$$

Thus, zero gradient columns fix coordinates to 0, columns equal to the bias gradient fix coordinates to 1, and identical nonconstant columns share one binary variable. The bias coordinate is retained as the final coordinate. After enumeration, compressed candidates are expanded by restoring fixed coordinates and copying representative values to duplicated coordinates. Under the rank condition, this compression preserves the candidate set while reducing the dimension of the binary candidate-enumeration problem. The equivalences hold because equal rank forces $\ker(\widetilde{S}^{\top}) = \ker(D)$, so the observable column relations of D are exactly the coordinate relations of $\widetilde{S}$. The proof is given in Appendix B

Batch-size inference. When the batch size is not known, it can be inferred in the full batch–temporal rank regime as

$$\widehat{B} = \frac{1}{T} \max_{\ell \in \mathcal{I}} \mathrm{rank}(D^{(\ell)}).$$

This inference requires full batch–temporal rank in at least one observed layer.

Candidate enumeration. Let $D_{\mathrm{c}} \in \mathbb{R}^{N\times d}$ be the compressed augmented gradient matrix. We compute a numerical row-space basis $R \in \mathbb{R}^{r\times d}$, estimating r from the observed singular values. Exact candidates satisfy $R^{\top}h = z$, with $h \in \mathbb{R}^r$, $z \in \{0,1\}^d$, and $z_d = 1$. In finite precision, we solve the mixed-integer linear program (MILP)

$$\begin{aligned} \min_{h,z,u} \quad & \mathbf{1}_d^{\top} u \\ \text{s.t.} \quad & -u \le R^{\top}h - z \le u, \\ & 0 \le u \le \epsilon_{\mathrm{eq}} \mathbf{1}_d, \\ & h \in \mathbb{R}^r, \quad z \in \{0,1\}^d, \quad z_d = 1. \end{aligned}$$

Each feasible binary pattern is excluded by a no-good cut before the next solve. Enumeration stops when the remaining MILP is proven infeasible or a computational limit is reached. Only infeasibility certifies exhaustion of the specified numerical problem. These pools are passed to Stage II. Numerical settings and stopping criteria are detailed in Appendix C.

2.3 Stage II: Reconstruction through Spatio-Temporal Consistency

Stage I provides candidate pools without temporal positions, occurrence counts, or example assignments. Stage II reconstructs the private batch by assigning input candidates from $\mathcal{P}_0$ to B trajectories of length T, using the known SNN dynamics and recovered downstream candidate sets as consistency constraints.

Stateful propagation. Examples evolve independently from the known initial neuronal states. For example a and layer ℓ, the forward dynamics are

$$i_{a,t}^{(\ell)} = W^{(\ell)} s_{a,t}^{(\ell-1)} + b^{(\ell)}, \qquad \big(v_{a,t}^{(\ell)}, s_{a,t}^{(\ell)}\big) = \Phi_\ell\big(v_{a,t-1}^{(\ell)}, i_{a,t}^{(\ell)}\big),$$

where $v_{a,t}^{(\ell)}$ contains the membrane and any additional neuronal state. The transition Φ_ℓ includes integration, thresholding, and reset. Consequently, the consistency of an input candidate depends on the preceding prefix through the carried states.

Consistency-guided search. We use depth-first search with backtracking, filling examples in a fixed order and their timesteps chronologically. Each extension propagates a candidate $x \in \mathcal{P}_0$ from a copy of the current example's neuronal states. Its induced spikes must belong to $\mathcal{P}_j$ at every constrained downstream representation j. Inconsistent extensions are rejected; retained extensions carry their updated states into the next timestep.

Backtracking restores states and occurrence counts and can revise earlier completed trajectories when the remaining batch has no feasible completion. Only input candidates are explicitly assigned; downstream sequences are induced by forward propagation.

Repeated patterns and rank checks. For each constrained representation j, including the input, let $r_j := \operatorname{rank}(D^{(j+1)})$. Since rank cannot exceed the number of distinct augmented spike vectors, $r_j = L$ rules out repetition. We therefore allow each pattern at most once across the batch when $r_j = L$, and permit reuse otherwise. A completed batch must additionally satisfy:

$$\operatorname{rank}\big(\widehat{\widetilde{S}}^{(j)}\big) \geq r_j$$

at every constrained representation, where $\widehat{\widetilde{S}}^{(j)}$ concatenates the predicted augmented spike vectors across all examples and timesteps. After q positions have been filled, a branch is pruned if $k_j + L - q < r_j$ for any j, where k_j counts the distinct patterns generated so far. Such a branch cannot attain the required terminal rank.

Conditional recovery. If the candidate pools contain all true spike patterns, the neuron dynamics and initial states are correct, and all pruning tests are sound, then the true batch remains feasible throughout Stage II and none of its prefixes is pruned. Since the search space over retained candidates is finite, exhaustive search without a computational cutoff eventually returns a feasible batch. Exact recovery is obtained when the dynamics and downstream-consistency constraints determine a unique feasible input batch, up to permutation of batch examples and the indistinguishability of repeated examples.

3 Experiments

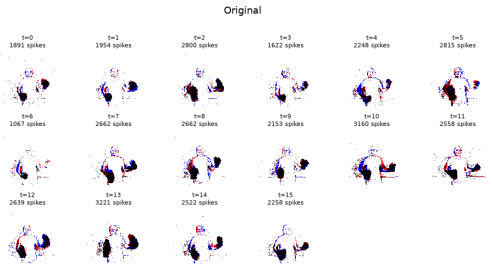
We evaluate reconstruction of private binary spike sequences from shared SNN gradients on four event-based vision datasets: N-MNIST, DVS128 Gesture, N-Caltech101, and CIFAR10-DVS. We first assess exact recovery across datasets, batch sizes, and temporal horizons. We then study two sources of practical difficulty: spike sparsity and gradient sparsity.

The main experiments use randomly initialized fully connected SNNs with dataset-specific dimensions. For N-MNIST, the network has layer dimensions $2312 - 512 - 256 - 128 - 10$, corresponding to an input layer, three hidden fully connected layers, and an output layer; dimensions for the other datasets are reported in Appendix E.1. We use leaky integrate-and-fire neurons with membrane decay $\beta = 0.9$, firing threshold 1.0, and the fast-sigmoid surrogate gradient implemented in `snnTorch`.

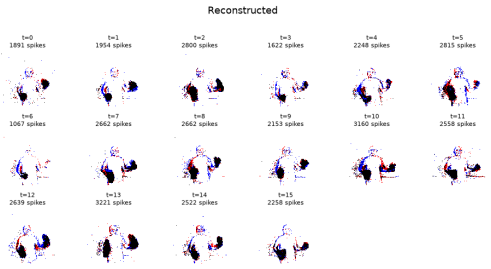
In all experiments, the reconstruction target is the binary spike sequence supplied to the first fully connected layer. Reconstruction accuracy is evaluated by comparing original and reconstructed spikes at matching timesteps. Since the attack is analytical, we use the percentage of exactly reconstructed sequences as the primary success metric. For visualization, we show temporal spike frames and temporally aggregated images, but aggregation is used only for display.

Table 1: Exact reconstruction across completed (T, B) configurations. Counts indicate exact reconstructions over completed configurations.

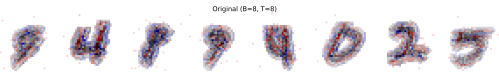
Dataset	T values	B values	Exact / completed
N-MNIST	$4, 6, 8$	$1, 2, 4, 8, 12, 16, 32$	21/21
DVS128 Gesture	$4, 6, 8$	$1, 2, 4, 8$	12/12
N-Caltech101	$4, 6, 8$	$1, 2, 4, 8$	12/12
CIFAR10-DVS	$4, 6, 8$	$1, 2, 4, 8$	12/12



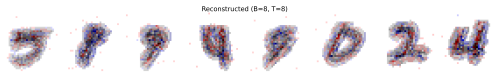
(a) DVS128 Gesture: original



(b) DVS128 Gesture: reconstructed



(c) N-MNIST: original



(d) N-MNIST: reconstructed

Figure 2: Qualitative reconstruction examples. The DVS128 Gesture example shows temporal spike reconstructions for $B = 1$ and $T = 16$, while the N-MNIST example shows a mini-batch reconstruction for $B = 8$ and $T = 8$, displayed as temporal aggregations for readability.

Computational budgets. Stage I enumerates binary row-space candidates using the guided MILP procedure from Section 2, with equality tolerance 10^{-8}, projection tolerance 10^{-7}, and a 3×10^3-second enumeration limit per layer. Stage II backtracking is limited to 2×10^6.

Experiments use one NVIDIA A100 80 GB GPU, 8 CPU cores, and 32 GB of memory per run.

3.1 Exact Recovery across Datasets and Parameter Settings

Table 1 summarizes reconstruction across datasets, batch sizes, and temporal horizons. The attack achieves exact recovery in all 57 completed configurations. In every completed configuration, the reconstructed spike sequence matches the original sequence exactly.

These results show that aggregation across both examples and timesteps does not prevent exact reconstruction in the evaluated rank-preserving settings. Additional configurations were not conducted due to time constraints (see Figure 3 for computational time).

Figure 2 shows representative temporal reconstructions for $B = 1$ and $T = 16$, illustrating that the attack recovers the spike sequence over time, and shows a representative N-MNIST mini-batch reconstruction for $B = 8$ and $T = 8$, displayed as temporal aggregations for readability. Additional reconstructions are reported in Appendix E.2.

3.2 Effect of the Batch Size

We further examine batch scaling on N-MNIST at $T = 8$, with batch sizes up to $B = 32$. Figure 3 reports the corresponding runtime and search effort. Exact reconstruction is achieved for all evaluated batch sizes, including $B = 32$, where $L = BT = 256$ which demonstrates the reconstruction ability of our approach against realistic and common batch sizes. Runtime increases substantially with L. In the largest setting, $B = 32, T = 8$, the complete run takes approximately 7.15×10^4 seconds.

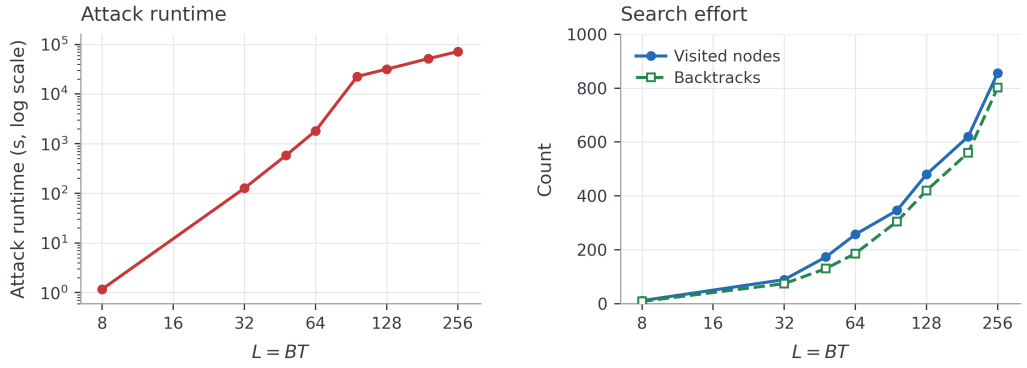


Figure 3: N-MNIST batch-scaling diagnostics for $T = 8$. The attack reconstructs exactly for all shown mini-batch sizes, up to $B = 32$ and $L = BT = 256$.

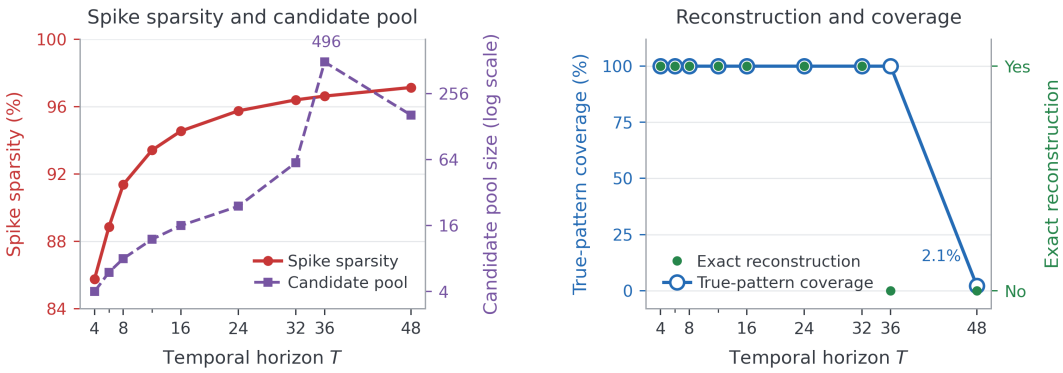


Figure 4: Temporal-horizon diagnostics on N-MNIST with $B = 1$: spike sparsity and candidate-pool size (left), and exact recovery and true-pattern coverage (right).

3.3 Effect of Spike and Gradient Sparsity

We examine how sparsity in the spike factor S and backward-signal factor G affects candidate recovery and reconstruction. Longer temporal horizons can produce sparse spike sequences, while deeper architectures can yield weak or sparse backward signals at early layers.

Figure 4 varies the temporal horizon on N-MNIST with $B = 1$. Spike sparsity and the candidate-pool size increase with T. Exact recovery succeeds through $T = 32$. At $T = 36$, true-pattern coverage remains complete, but the attack does not return a complete reconstruction within the configured Stage II search budget. At $T = 48$, retained coverage is incomplete, so the search lacks some true input patterns.

Figure 5 examines first-layer gradient sparsity across tested fully connected settings on N-MNIST with $B = 1$ and $T = 8$. The first-layer gradient factor becomes sparser in the deeper tested settings. Exact recovery holds when retained true-pattern coverage is complete and fails in the deepest settings, where coverage drops. This experiment is used only to study how first-layer gradient sparsity affects the attack.

Together, these results highlight the difference between candidate coverage and practical recoverability. Rank preservation characterizes when the true spike patterns belong to the exact candidate space, but it does not guarantee that they are uniquely or easily identifiable. When spikes or backward signals are sparse, the row-space constraints can admit many spurious binary candidates. The retained pool may therefore contain the true patterns while also containing false candidates that enlarge the search space. Stage II may then exhaust its budget before reaching the true assignment.

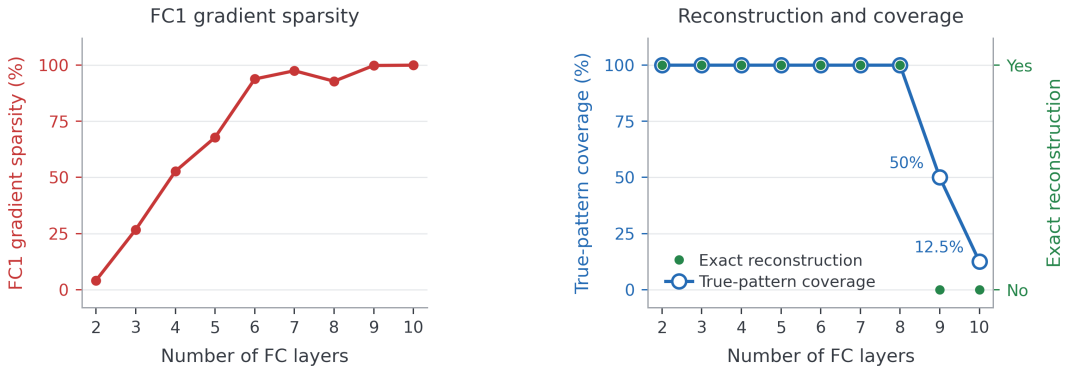


Figure 5: Gradient-sparsity diagnostics on N-MNIST with $B = 1$ and $T = 8$: first-layer gradient-factor sparsity (left), and exact recovery and true-pattern coverage (right).

4 Related Work

Gradient inversion in ANNs. Gradient leakage was first observed at the level of a single fully connected layer, where the weight and bias gradients of a neuron determine its input up to a scalar (Phong et al., 2018). Zhu et al. (2019) and Zhao et al. (2020) turned this observation into a general attack by optimizing a dummy input to match the observed gradient. Geiping et al. (2020) used cosine similarity and added image priors, while Ren et al. (2022) substituted a generative prior for pixel-space optimization. These attacks are approximate: their success is measured by perceptual similarity and depends on initialization, priors, and batch size (Huang et al., 2021).

Analytical inversion. A second line of work recovers inputs analytically rather than by optimizing dummy data. R-GAP (Zhu & Blaschko, 2021) recursively solves layerwise linear systems, but is limited to single-example reconstruction. Recent exact-inversion attacks further show that gradient leakage can extend beyond single-example reconstruction: SPEAR (Dimitrov et al., 2024) uses low-rank weight-gradient structure and ReLU-induced sparsity to reconstruct batches in fully connected networks, while DAGER (Petrov et al., 2024) extends exact inversion to large language models.

Our setting differs in the structure that makes inversion possible. Rather than exploiting ReLU sparsity or transformer-specific structure, we study the SNN-specific factorization of gradients into a temporal gradient factor and a binary presynaptic spike factor. DAGER is closest in spirit, since it tests discrete candidates against a gradient-induced span; however, it enumerates a finite token vocabulary under embedding-dimension constraints. In contrast, an M-neuron spike pattern lies in a Boolean space with up to 2^M possible candidates, and temporal aggregation introduces BT example–timestep contributions. SNN inversion therefore requires constrained binary candidate recovery together with dynamics-aware temporal assignment.

FL and private SNNs. Several studies report that SNNs resist attacks that succeed against ANNs. Aksu et al. (2025) adapt DLG, iDLG, and GRNN to the spiking domain and report substantially lower attack success than on matched ANNs. They attribute this gap to non-differentiable spike generation and to temporal dynamics that better decouple the gradient from the individual input. Moshruba et al. (2025) report an analogous advantage against membership inference and relate it to the choice of surrogate gradient and to quantization.

We view these lower attack-success results as evidence of attack mismatch, rather than evidence that SNN gradients lack reconstructive information. The adapted attacks are optimization-based and rely on objectives designed for conventional ANN gradients; temporal aggregation over T timesteps changes this structure and can degrade attacks that do not exploit spiking computation. Distinguishing attack mismatch from genuine protection therefore requires an attack built on the algebraic structure of SNN gradients. We construct such an attack and show that temporal aggregation converts recovery into a finite constrained problem over binary spike patterns, rather than removing the private input from the gradient.

AI USE STATEMENT

In this work, we used LLM-based assistants only for language polishing and grammar checking of the manuscript. They were not used to generate research ideas, derive theoretical results, design experiments, or analyse data. The authors reviewed all edited text and take full responsibility for the content of the paper.

ETHICS STATEMENT

This work studies privacy leakage from shared gradients in spiking neural networks (SNNs), with particular emphasis on federated learning (FL). We view the primary ethical implication of this work as highlighting a previously underexplored privacy risk rather than enabling access to data that would otherwise be publicly available.

Our experiments are conducted on established event-based vision benchmarks (N-MNIST, DVS128 Gesture, N-Caltech101, and CIFAR10-DVS). No personally or otherwise sensitive private data was collected.

We believe that making the leakage mechanism explicit is important for evaluating the privacy assumptions of SNN-based federated learning and for motivating appropriate defenses that reduce exploitable leakage.

REPRODUCIBILITY STATEMENT

The anonymous repository containing the implementation of SpikeLeak, the experimental code, and the configurations required to reproduce the experiments reported in this paper is available at `https://anonymous.4open.science/r/SpikeLeak-A4D5/`. The repository includes reproducing the candidate-recovery and spatio-temporal reconstruction experiments, together with the evaluation procedures used to measure exact reconstruction, candidate coverage, rank, search effort, and runtime.

REFERENCES

Dogukan Aksu, Jesus Martinez del Rincon, and Ihsen Alouani. Privacy in federated learning with spiking neural networks. *arXiv preprint arXiv:2511.21181*, 2025.

Sauptik Dhar, Junyao Guo, Jiayi Liu, Samarth Tripathi, Unmesh Kurup, and Mohak Shah. A survey of on-device machine learning: An algorithms and learning theory perspective. *ACM Transactions on Internet of Things*, 2(3):1–49, 2021.

Dimitar I. Dimitrov, Maximilian Baader, Mark Niklas Müller, and Martin Vechev. SPEAR: Exact gradient inversion of batches in federated learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients: How easy is it to break privacy in federated learning? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Sukhpal Singh Gill, Muhammed Golec, Jianmin Hu, Minxian Xu, Junhui Du, Huaming Wu, Guneet Kaur Walia, Subramaniam Subramanian Murugesan, Babar Ali, Mohit Kumar, et al. Edge ai: A taxonomy, systematic review and future directions. *Cluster Computing*, 28(1):18, 2025.

Yangsibo Huang, Samyak Gupta, Zhao Song, Kai Li, and Sanjeev Arora. Evaluating gradient inversion attacks and defenses in federated learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Tian Liu, Jun Xia, Xin Fu, Shui Yu, and Mingsong Chen. Efficient federated learning for aiot applications using knowledge distillation. *IEEE Internet of Things Journal*, 10(8):7229–7242, 2023. IEEE Xplore: `https://ieeexplore.ieee.org/document/9987477/` — arXiv: `https://arxiv.org/abs/2111.14347`.

Lingjuan Lyu, Han Yu, Xingjun Ma, Chen Chen, Lichao Sun, Jun Zhao, Qiang Yang, and Philip S Yu. Privacy and robustness in federated learning: Attacks and defenses. *IEEE transactions on neural networks and learning systems*, 35(7):8726–8746, 2022.

Paul A. Merolla, John V. Arthur, Rodrigo Alvarez-Icaza, Andrew S. Cassidy, Jun Sawada, Filipp Akopyan, Bryan L. Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, Bernard Brezzo, Ivan Vo, Steven K. Esser, Rathinakumar Appuswamy, Brian Taba, Arnon Amir, Myron D. Flickner, William P. Risk, Rajit Manohar, and Dharmendra S. Modha. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, 2014. doi: 10.1126/science.1254642.

Ayana Moshruba, Shay Snyder, Hamed Poursiami, and Maryam Parsa. On the privacy-preserving properties of spiking neural networks with unique surrogate gradients and quantization levels. *arXiv preprint arXiv:2502.18623*, 2025.

Ivo Petrov, Dimitar I Dimitrov, Maximilian Baader, Mark N Müller, and Martin Vechev. Dager: Exact gradient inversion for large language models. *Advances in neural information processing systems*, 37:87801–87830, 2024.

Le Trieu Phong, Yoshinori Aono, Takuya Hayashi, Lihua Wang, and Shiho Moriai. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Transactions on Information Forensics and Security*, 13(5):1333–1345, 2018.

Hanchi Ren, Jingjing Deng, and Xianghua Xie. GRNN: Generative regression neural network. a data leakage attack for federated learning. *ACM Transactions on Intelligent Systems and Technology*, 13(4), 2022.

Nicola Rieke, Jonny Hancox, Wenqi Li, Fausto Milletari, Holger R Roth, Shadi Albarqouni, Spyridon Bakas, Mathieu N Galtier, Bennett A Landman, Klaus Maier-Hein, et al. The future of digital health with federated learning. *NPJ digital medicine*, 3(1):119, 2020.

Nicolas Skatchkovsky, Hyeryung Jang, and Osvaldo Simeone. Federated neuromorphic learning of spiking neural networks for low-power edge intelligence. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.

Guangzhi Tang, Arpit Shah, and Konstantinos P. Michmizos. Spiking neural network on neuromorphic hardware for energy-efficient unidimensional slam. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4176–4181, 2019. doi: 10.1109/IROS40897.2019.8967864.

Yeshwanth Venkatesha, Youngeun Kim, Leandros Tassiulas, and Priyadarshini Panda. Federated learning with spiking neural networks. *IEEE Transactions on Signal Processing*, 69:6183–6194, 2021.

Helin Yang, Kwok-Yan Lam, Liang Xiao, Zehui Xiong, Hao Hu, Dusit Niyato, and H. Vincent Poor. Lead federated neuromorphic learning for wireless edge artificial intelligence. *Nature Communications*, 13(4269), 2022.

Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. iDLG: Improved deep leakage from gradients. *arXiv preprint arXiv:2001.02610*, 2020.

Zhi Zhou, Xu Chen, En Li, Liekang Zeng, Ke Luo, and Junshan Zhang. Edge intelligence: Paving the last mile of artificial intelligence with edge computing. *Proceedings of the IEEE*, 107(8): 1738–1762, 2019.

Junyi Zhu and Matthew B. Blaschko. R-GAP: Recursive gradient attack on privacy. In *International Conference on Learning Representations (ICLR)*, 2021.

Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

A PROOFS FOR STAGE I

A.1 PROOF OF PROPOSITION 1

For a fully connected layer, the affine current at example–timestep position q is

$$i_q = W s_q + b,$$

where $s_q \in \{0,1\}^M$. Let $g_q = \partial \mathcal{L}/\partial i_q \in \mathbb{R}^N$ denote the total backward signal at i_q, including the temporal dependencies propagated through backpropagation through time.

The contribution of position q to the weight gradient is

$$\frac{\partial \mathcal{L}}{\partial W}\Big|_q = g_q s_q^\top,$$

and the contribution to the bias gradient is

$$\frac{\partial \mathcal{L}}{\partial b}\Big|_q = g_q.$$

Summing over all $L = BT$ example–timestep positions gives

$$\nabla W = \sum_{q=1}^{L} g_q s_q^\top, \qquad \nabla b = \sum_{q=1}^{L} g_q.$$

If the gradients are averaged rather than summed, the averaging factor can be absorbed into the definition of G. With

$$S = [s_1, \ldots, s_L] \in \{0,1\}^{M \times L}, \qquad G = [g_1, \ldots, g_L] \in \mathbb{R}^{N \times L},$$

we obtain

$$\nabla W = G S^\top, \qquad \nabla b = G \mathbf{1}_L.$$

Therefore,

$$D := [\nabla W \mid \nabla b] = G \begin{bmatrix} S \\ \mathbf{1}_L^\top \end{bmatrix}^\top = G \widetilde{S}^\top,$$

where

$$\widetilde{S} := \begin{bmatrix} S \\ \mathbf{1}_L^\top \end{bmatrix} \in \{0,1\}^{(M+1) \times L}.$$

This proves Proposition 1.

A.2 PROOF OF THEOREM 1

Recall that

$$D = G \widetilde{S}^\top.$$

Taking transposes gives

$$D^\top = \widetilde{S} G^\top.$$

Therefore every column of $D^\top$ is a linear combination of the columns of $\widetilde{S}$, and hence

$$\mathcal{R}(D) := \mathrm{col}(D^\top) \subseteq \mathrm{col}(\widetilde{S}).$$

If

$$\mathrm{rank}(D) = \mathrm{rank}(\widetilde{S}),$$

then the two spaces have the same dimension. Since one is contained in the other, they must be equal:

$$\mathcal{R}(D) = \mathrm{col}(\widetilde{S}).$$

We now characterize the binary candidates. By definition,

$$\mathcal{C}(D) = \left\{ x \in \{0,1\}^M : [x^\top, 1]^\top \in \mathcal{R}(D) \right\}.$$

Under the rank condition, $\mathcal{R}(D) = \mathrm{col}(\widetilde{S})$. Thus,

$$[x^\top, 1]^\top \in \mathcal{R}(D)$$

if and only if there exists $\alpha \in \mathbb{R}^L$ such that

$$[x^\top, 1]^\top = \widetilde{S}\alpha = \begin{bmatrix} S \\ \mathbf{1}_L^\top \end{bmatrix} \alpha.$$

This is equivalent to

$$x = S\alpha, \qquad \mathbf{1}_L^\top \alpha = 1.$$

The condition $\mathbf{1}_L^\top \alpha = 1$ exactly states that x lies in the affine hull of the columns $s_1, \ldots, s_L$. Therefore,

$$\mathcal{C}(D) = \{0,1\}^M \cap \mathrm{aff}\{s_1, \ldots, s_L\}.$$

Since each true spike vector s_q is one of the points generating this affine hull and is binary, it follows that

$$\{s_q : 1 \le q \le L\} \subseteq \mathcal{C}(D).$$

Finally, if $\mathrm{rank}(G) = L$, then G has full column rank and admits a left inverse. Multiplication by G therefore preserves the rank of $\widetilde{S}^\top$, so

$$\mathrm{rank}(D) = \mathrm{rank}(G\widetilde{S}^\top) = \mathrm{rank}(\widetilde{S}^\top) = \mathrm{rank}(\widetilde{S}).$$

Thus, $\mathrm{rank}(G) = L$ is a sufficient condition for the theorem's rank-preservation condition.

A.3 Rank Preservation and Binary Candidate Coverage

This appendix explains how the ranks of the gradient factor, spike factor, and observed gradient determine whether Stage I includes every true spike pattern. We distinguish redundancy in the spike patterns from additional information loss through the gradient factor, and explain their different consequences for reconstruction.

A.3.1 Setup and overview

For a selected fully connected layer, let $L = BT$ and write

$$D = G\widetilde{S}^\top, \qquad G \in \mathbb{R}^{N \times L}, \qquad \widetilde{S} = \begin{bmatrix} S \\ \mathbf{1}_L^\top \end{bmatrix} \in \{0,1\}^{(M+1)\times L}.$$

The columns of S are the presynaptic spike vectors $s_1, \ldots, s_L$ over all example–timestep positions. Define

$$r_D := \mathrm{rank}(D), \qquad r_S := \mathrm{rank}(\widetilde{S}), \qquad r_G := \mathrm{rank}(G).$$

Here, r_S refers to the *augmented* spike matrix, whose rank can exceed that of S by one. The observed row space and exact candidate set are

$$\mathcal{R}(D) := \mathrm{col}(D^\top), \qquad \mathcal{C}(D) := \left\{x \in \{0,1\}^M : [x^\top, 1]^\top \in \mathcal{R}(D)\right\}.$$

Throughout this appendix, ranks and subspace membership are exact unless stated otherwise. *Candidate coverage* means

$$\{s_1, \ldots, s_L\} \subseteq \mathcal{C}(D).$$

This property does not establish that numerical enumeration has finished or that the input is uniquely reconstructible. Table 2 summarizes the three possible regimes.

A.3.2 Exact characterization of candidate coverage

Let

$$\mathcal{U} := \mathrm{col}(\widetilde{S}^\top) \subseteq \mathbb{R}^L.$$

This is the subspace on which multiplication by G acts. It is distinct from $\mathrm{col}(\widetilde{S}) \subseteq \mathbb{R}^{M+1}$, which is the span of the augmented spike vectors.

Table 2: Rank regimes for exact binary candidate recovery, with $L = BT$. The regimes are exhaustive because $r_D \leq r_S \leq L$. Coverage concerns the exact set $\mathcal{C}(D)$, not numerical enumeration or uniqueness of the reconstructed input.

Regime	Rank condition	Stage-I coverage	Implication for Stage II
Rank equal to L	$r_D = r_S = L$ (hence $r_G = L$)	All true patterns are included and distinct; additional binary candidates may still exist.	A pattern occurs at most once at this representation. Candidate selection, example assignment, and temporal order remain to be determined.
Rank preservation with spike redundancy	$r_D = r_S < L$	All true patterns are included. Repetitions or affine dependencies reduce the spike rank.	Permit reuse because repetition is possible. Dynamics and downstream constraints restrict occurrences and assignments.
Rank loss through the gradient factor	$r_D < r_S$	At least one true pattern is absent; some others may remain included.	Reuse cannot restore missing patterns. The true input is excluded if this exact pool is enforced as a hard constraint.

Proposition 2 (Coverage and rank preservation)**.** *The following conditions are equivalent:*

$$\begin{aligned}&\{s_1, \dots, s_L\} \subseteq \mathcal{C}(D),\\ &\mathcal{R}(D) = \mathrm{col}(\widetilde{S}),\\ &r_D = r_S,\\ &\mathcal{U} \cap \ker(G) = \{0\}.\end{aligned}$$

More precisely,

$$r_D = r_S - \dim\big(\mathcal{U} \cap \ker(G)\big). \tag{1}$$

Under these equivalent conditions,

$$\mathcal{C}(D) = \{0,1\}^M \cap \mathrm{aff}\{s_1, \dots, s_L\}. \tag{2}$$

Proof. Since $D^\top = \widetilde{S} G^\top$,

$$\mathcal{R}(D) \subseteq \mathrm{col}(\widetilde{S}).$$

If every true pattern belongs to $\mathcal{C}(D)$, then every column of $\widetilde{S}$ belongs to $\mathcal{R}(D)$. Their span therefore gives the reverse inclusion. Conversely, equality of the two spaces includes every augmented true vector. Since the spaces are nested, they are equal if and only if their dimensions r_D and r_S agree.

The image of the restriction of G to $\mathcal{U}$ is

$$\mathrm{col}(G\widetilde{S}^\top) = \mathrm{col}(D).$$

Rank–nullity for this restriction yields equation 1, proving the remaining equivalence.

Under rank preservation, $[x^\top, 1]^\top \in \mathcal{R}(D)$ holds exactly when

$$[x^\top, 1]^\top = \widetilde{S}\alpha$$

for some $\alpha \in \mathbb{R}^L$. Its last coordinate requires $\mathbf{1}_L^\top \alpha = 1$, and its first M coordinates require $x = S\alpha$. Intersecting these affine combinations with $\{0,1\}^M$ gives equation 2. □

Thus, full column rank of G is sufficient for coverage because $\ker(G) = \{0\}$. It is not necessary: a rank-deficient G can still be injective on $\mathcal{U}$. In particular, $r_G \geq r_S$ is necessary for coverage, but is not sufficient; the subspace annihilated by G also matters.

A.3.3 THE THREE RANK REGIMES

Rank equal to L. If $r_D = r_S = L$, then $r_G = L$, and the L augmented spike vectors are linearly independent. Repeated spike vectors would give identical augmented columns and violate this independence. Hence, all true patterns are distinct and included in $\mathcal{C}(D)$. This does not exclude additional binary points in their affine hull.

For example, let $L = 3$, $M = 3$, $G = I_3$, and

$$s_1 = (1, 0, 1)^\top, \qquad s_2 = (0, 1, 1)^\top, \qquad s_3 = (0, 0, 1)^\top.$$

Then $r_D = r_S = r_G = 3$, but

$$x = s_1 + s_2 - s_3 = (1, 1, 1)^\top$$

is an additional binary candidate. The coefficients sum to one, so x lies in the affine hull. Rank equal to L therefore guarantees coverage, not unique candidate selection or unique reconstruction.

Rank preservation with spike redundancy. If $r_D = r_S < L$, the same coverage guarantee holds. Repetition is one possible source of the lower rank. For example, let

$$u = (0, 0)^\top, \qquad v = (1, 0)^\top, \qquad S = [u, v, u], \qquad G = I_3.$$

Then $r_D = r_S = 2 < L = 3$ and $\mathcal{C}(D) = \{u, v\}$. The set records u once although it occurs twice in the sequence. Consuming it permanently after its first use would exclude the true input.

Lower rank can also occur without any repeated pattern. Take $G = I_4$ and

$$s_1 = (0, 0)^\top, \quad s_2 = (1, 0)^\top, \quad s_3 = (0, 1)^\top, \quad s_4 = (1, 1)^\top.$$

All four patterns are distinct, but

$$\widetilde{s}_4 = \widetilde{s}_2 + \widetilde{s}_3 - \widetilde{s}_1,$$

so $r_D = r_S = 3 < L = 4$. The observed lower rank therefore cannot by itself establish whether repetition occurs or determine occurrence counts.

More generally, if k distinct patterns occur among the L positions, then

$$r_D \le r_S \le k \le L. \tag{3}$$

Consequently, $r_D = L$ rules out repetition. When $r_D < L$, permitting reuse preserves the possibility of repeated occurrences without asserting that every candidate repeats. Stage II applies this decision separately at each constrained representation and counts occurrences across all examples and timesteps.

Rank loss through the gradient factor. If $r_D < r_S$, the inclusion

$$\mathcal{R}(D) \subsetneq \mathrm{col}(\widetilde{S})$$

is strict. At least one augmented true vector must lie outside $\mathcal{R}(D)$: otherwise their span would be contained in $\mathcal{R}(D)$, contradicting strict inclusion. Thus, at least one true pattern is absent from $\mathcal{C}(D)$. Others may remain included.

If $r_G < r_S$, then rank loss necessarily occurs. However, this condition is sufficient but not necessary: rank loss can also occur when $r_G \ge r_S$, depending on how $\ker(G)$ intersects the spike span. For example, take $M = 1$, $L = 3$, and

$$S = [0 \quad 1 \quad 0], \qquad \widetilde{S}^\top = \begin{bmatrix} 0 & 1 \\ 1 & 1 \\ 0 & 1 \end{bmatrix}, \qquad G = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}.$$

Although $r_G = r_S = 2$,

$$D = G\widetilde{S}^\top = \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix}, \qquad r_D = 1, \qquad \mathcal{C}(D) = \{1\}.$$

The true pattern 1 remains included, while the true pattern 0 is missing. In contrast, choosing

$$G' = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

gives $\mathrm{rank}(G') = 2$ and $\mathrm{rank}(G'\widetilde{S}^\top) = 2$, restoring coverage. This shows why the rank of G alone does not determine rank preservation.

These examples illustrate the factorization algebra; they do not assert that every chosen G is realized by a particular trained SNN.

A.3.4 Consequences for reconstruction and numerical computation

What Stage II must determine. In either rank-preserving regime, Stage I includes all true patterns but may also include additional ones. It supplies no example assignments, temporal positions, or occurrence counts. Stage II selects and assigns first-layer candidates to example trajectories, carrying each example's neuronal state through time. The induced downstream spikes must satisfy the corresponding recovered candidate constraints.

Reuse accommodates possible repetitions; consistency constrains which occurrences and orders are feasible. Exact reconstruction additionally requires the true patterns to be retained by numerical enumeration and the search to identify the true input among the feasible sequences.

If a true input pattern is absent from the input pool, selection from that pool cannot reconstruct its occurrence. Likewise, if a required downstream pattern is absent from a pool enforced as a hard membership constraint, replaying the true input fails that constraint. These statements concern the specified exact candidate pools; they do not establish impossibility for other reconstruction methods or for a different selection of constraints.

Dimensional restrictions. Since $r_D \le N$ and $r_S \le \min\{M+1, L\}$, coverage requires $r_S \le N$. When $r_S = L$, preserving this rank therefore requires

$$L = BT \le \min\{N, M+1\}.$$

These dimensions are necessary, not sufficient: the factors must also preserve rank. When $L > N$, rank L cannot be preserved, but a lower-rank spike matrix may still satisfy $r_D = r_S$.

Conversely, if $r_D = M+1$, then

$$\mathcal{R}(D) = \mathbb{R}^{M+1}, \qquad \mathcal{C}(D) = \{0,1\}^M.$$

Coverage then places no restriction on the binary patterns.

Exact coverage versus numerical recovery. The results above concern the exact factorization and exact row space. Observed rank $r_D = L$ certifies rank preservation in exact arithmetic when L is known. A smaller observed rank alone cannot distinguish spike redundancy from additional rank loss through G.

In finite precision, singular-value thresholds, approximate compression, membership tolerances, and enumeration limits can change the returned pool. A solver's infeasibility certificate exhausts the specified numerical enumeration problem; it does not by itself certify equality with the exact set $\mathcal{C}(D)$. Gradient-rank estimation is asymmetric in Stage II: overestimating a rank can incorrectly forbid repetition or reject the true input, whereas underestimating it mainly weakens pruning and may admit additional feasible sequences. These numerical issues must be distinguished from the algebraic regimes described here.

B Proof of Lossless Coordinate Compression

Assume the rank condition of Theorem 1:

$$\operatorname{rank}(D) = \operatorname{rank}(\widetilde{S}).$$

Let $A := \widetilde{S}^\top$, so that $D = GA$. For each augmented coordinate j, write

$$\tau_j := \widetilde{S}^\top_{j,:} = Ae_j \in \{0,1\}^L, \qquad d_j = De_j,$$

where e_j is the j-th standard basis vector in $\mathbb{R}^{M+1}$. Thus, τ_j records augmented coordinate j across the L example–timestep positions. In particular, $\tau_{M+1} = \mathbf{1}_L$ and $d_{M+1} = \nabla b$.

Proof. We first establish the column relations, then show that the compressed and original candidate sets correspond bijectively.

Column relations. Since $D = GA$,

$$\ker(A) \subseteq \ker(D).$$

Both matrices have $M + 1$ columns and, by assumption, the same rank. Their null spaces therefore have the same dimension, giving

$$\ker(A) = \ker(D).$$

Applying this equality to e_j, $e_j - e_{M+1}$, and $e_j - e_k$, respectively, yields, for $j, k \le M$,

$$\begin{aligned} d_j = 0 &\iff \tau_j = 0, \\ d_j = d_{M+1} &\iff \tau_j = \tau_{M+1} = \mathbf{1}_L, \\ d_j = d_k &\iff \tau_j = \tau_k. \end{aligned}$$

Thus, the observed column relations identify coordinates that are always zero, always one, or identical across all L example–timestep positions. Moreover, $\tau_{M+1} \neq 0$ implies $d_{M+1} \neq 0$, so the zero-column and bias-column cases cannot overlap.

Preservation of the candidate set. Construct $D_{\mathrm{c}} \in \mathbb{R}^{N \times d}$ by retaining one representative from each group of identical nonconstant columns and keeping the bias column last.

Let $\Pi : \mathbb{R}^{M+1} \to \mathbb{R}^d$ select these retained coordinates. Define the linear expansion map $H : \mathbb{R}^d \to \mathbb{R}^{M+1}$ as follows: zero coordinates are restored to zero, coordinates equal to the bias coordinate receive the final compressed coordinate, and duplicated coordinates receive their representative's value. The augmented bias coordinate also receives the final compressed coordinate.

By construction,

$$D = D_{\mathrm{c}} H^\top, \qquad \Pi H = I_d.$$

Consequently,

$$\mathcal{R}(D) = \mathrm{col}(D^\top) = H\,\mathrm{col}(D_{\mathrm{c}}^\top).$$

Define the compressed augmented candidate set

$$\mathcal{C}_{\mathrm{c}} := \left\{ z \in \{0,1\}^d : z_d = 1,\ z \in \mathrm{col}(D_{\mathrm{c}}^\top) \right\}.$$

For every $z \in \mathcal{C}_{\mathrm{c}}$, the expanded vector Hz is binary, has final coordinate one, and belongs to $\mathcal{R}(D)$. Removing its final coordinate therefore gives a candidate in $\mathcal{C}(D)$.

Conversely, let $x \in \mathcal{C}(D)$ and write $\widetilde{x} = [x^\top, 1]^\top$. The row-space identity implies $\widetilde{x} = Hy$ for some $y \in \mathrm{col}(D_{\mathrm{c}}^\top)$. Applying Π gives

$$y = \Pi \widetilde{x}.$$

Since Π selects coordinates and retains the bias coordinate last, y is binary and $y_d = 1$. Hence $y \in \mathcal{C}_{\mathrm{c}}$.

Finally, $\Pi H = I_d$ makes expansion injective. Thus,

$$\left\{ [x^\top, 1]^\top : x \in \mathcal{C}(D) \right\} = H\mathcal{C}_{\mathrm{c}},$$

with a one-to-one correspondence between the two candidate sets. Compression is therefore lossless in exact arithmetic. □

C Numerical Candidate Enumeration

This appendix specifies the numerical row-space construction, candidate verification, and stopping rules used by the reference implementation. Table 3 summarizes the configuration.

Numerical rank and row-space bases. For each matrix $A \in \{D_{\mathrm{c}}, D\}$, compute $A = U_A \Sigma_A V_A^\top$ and retain singular values satisfying

$$\sigma_i(A) > \tau_A, \qquad \tau_A := \max\{\tau_{\mathrm{abs}}, \tau_{\mathrm{rel}} \sigma_1(A)\}.$$

The numerical rank is

$$r_A := \#\{i : \sigma_i(A) > \tau_A\},$$

and R_A contains the corresponding rows of $V_A^\top$. We estimate the compressed and full ranks separately rather than fixing either to $L = BT$.

By default, $\tau_{\text{abs}} = 0$ and

$$\tau_{\text{rel}} = \max\{m_A, n_A\}\, \varepsilon_{\text{mach}}(A),$$

where m_A, n_A are the dimensions of A, and $\varepsilon_{\text{mach}}(A)$ is the machine precision of its original floating-point dtype. SVD is evaluated in double precision; this conversion does not recover precision absent from the original gradients. Singular values close to the threshold make the estimated rank sensitive to numerical error.

Approximate coordinate compression. A column d_j is treated as zero when $\|d_j\|_2 \le \tau_0$. Equality to the bias column and equality between potential duplicate columns use the test

$$\|a - b\|_2 \le \eta_{\text{abs}} + \eta_{\text{rel}} \max\{\|a\|_2, \|b\|_2, 1\}.$$

Potential duplicates are first grouped by rounding their gradient columns to 12 decimal places, then checked against a representative using this norm test. The bias coordinate is retained last. These tolerance-based tests approximate the exact compression rules; the full-space verification below provides an additional check after expansion.

MILP residual and binary constraints. The MILP keeps z binary and fixes $z_d = 1$, while $h \in \mathbb{R}^{r_{D_c}}$ remains unrestricted. The tolerance ϵ_{eq} bounds the discrepancy between the continuous row-space representation and the binary vector:

$$\|R_{D_c}^\top h - z\|_\infty \le \epsilon_{\text{eq}}.$$

It does not make the spike values fractional. We minimize the sum of the nonnegative coordinate slacks, as specified in the main text. The reference configuration uses no random perturbation of this objective.

The implementation calls `scipy.optimize.milp` and requests a relative MIP gap of zero. Solver feasibility and integrality tolerances are separate from ϵ_{eq}; returned integer variables are rounded before residual evaluation.

Candidate verification. For an orthonormal row-space basis R_A, define the projection residual

$$\rho_A(v) := \|v - R_A^\top R_A v\|_2.$$

A compressed candidate must satisfy

$$\rho_{D_c}(z) \le \epsilon_{\text{c}}.$$

After expansion to the augmented vector $\widetilde{x}$, it must also satisfy

$$\rho_D(\widetilde{x}) \le \epsilon_{\text{f}}.$$

Both thresholds are absolute Euclidean residual tolerances. The full-space check matters because approximate coordinate compression can alter the recovered subspace.

The equality and projection tolerances constrain different residuals. For an exactly feasible MILP solution and an exact orthonormal basis,

$$\rho_{D_c}(z) \le \|R_{D_c}^\top h - z\|_2 \le \sqrt{d}\, \epsilon_{\text{eq}}.$$

To avoid rejecting candidates that satisfy the MILP residual bound solely because the compressed dimension d is large, the reference setting uses

$$\epsilon_{\text{c}} = \max\{10^{-7}, \sqrt{d}\, \epsilon_{\text{eq}}\}.$$

Equality residuals are recomputed for diagnostics; acceptance uses the two projection checks above.

Enumeration and stopping rules. After each returned pattern $\bar{z}$, we add the no-good cut

$$\sum_{j:\bar{z}_j=0} z_j + \sum_{j:\bar{z}_j=1} (1 - z_j) \geq 1.$$

This excludes precisely that binary vector while allowing every other binary vector. The cut is added even if the pattern fails subsequent verification, preventing repeated enumeration of rejected patterns.

Each layer has a maximum number of enumeration iterations, a per-solve time limit, and a total enumeration time budget. Each solve is limited by the smaller of its per-solve limit and the remaining enumeration budget.

Enumeration ends when the solver reports infeasibility, a configured limit is reached, or another solver failure occurs. Only the infeasibility outcome certifies exhaustion of the configured numerical MILP. The stopping reason and completeness flag are retained separately. A residual-ranked subset of the retained pool does not inherit the enumeration completeness claim.

Algorithm 2 Stateful consistency-guided search

function SEARCH($\{\mathcal{P}_j\}_{j\in\mathcal{K}}, \theta, V_0, \{r_j\}_{j\in\mathcal{K}}, B, T, Q$)
  $L \leftarrow BT$, $\nu \leftarrow 0$
  $c_j \leftarrow 1$ if $r_j = L$, and $c_j \leftarrow L$ otherwise, for every $j \in \mathcal{K}$
  Initialize empty batch X and counts $n \leftarrow 0$
  $V_a \leftarrow \text{COPY}(V_0)$ for $a = 1, \ldots, B$
  return BACKTRACK$(0, X, V, n)$
end function
function BACKTRACK(q, X, V, n)
  if $q = L$ **then**
    if $\operatorname{rank}(\widehat{\widehat{S}}^{(j)}(X)) \geq r_j$ for every $j \in \mathcal{K}$ **then**
      return $(X, \text{FEASIBLE})$
    end if
    return $(\varnothing, \text{EXHAUSTED})$
  end if
  Let (a, t) be the next position in the fixed schedule
  for all $x \in \mathcal{P}_0$ **do**
    if $\nu \geq Q$ **then**
      return $(\varnothing, \text{BUDGET})$
    end if
    $\nu \leftarrow \nu + 1$
    $(X', V', n') \leftarrow \text{COPY}(X, V, n)$
    $(V'_a, \{y_j\}_{j\in\mathcal{J}}) \leftarrow \text{FORWARDSTEP}(\theta, V'_a, x)$
    $y_0 \leftarrow x, \quad X'_{a,t} \leftarrow x$
    $n'_j(y_j) \leftarrow n_j(y_j) + 1$ for every $j \in \mathcal{K}$
    $k'_j \leftarrow |\{p : n'_j(p) > 0\}|$ for every $j \in \mathcal{K}$
    if $y_j \in \mathcal{P}_j$ for every $j \in \mathcal{J}$, $n'_j(y_j) \leq c_j$ and $k'_j + L - q - 1 \geq r_j$ for every $j \in \mathcal{K}$ **then**
      $(\widehat{X}, \sigma) \leftarrow \text{BACKTRACK}(q + 1, X', V', n')$
      if $\sigma \neq \text{EXHAUSTED}$ **then**
        return $(\widehat{X}, \sigma)$
      end if
    end if
  end for
  return $(\varnothing, \text{EXHAUSTED})$
end function

Scope of numerical verification. Small residuals establish proximity to the estimated row spaces, not membership in the exact row spaces. Likewise, exhausting the numerical MILP certifies only that no further binary vector satisfies the configured numerical constraints; it does not by itself guarantee that every true spike pattern was retained. Rank estimation, approximate compression, and residual filtering can exclude true patterns, while tolerances can admit additional ones.

Table 3: Numerical settings of the reference candidate-enumeration configuration. The time budgets apply separately to each selected layer.

Setting	**Value**
SVD arithmetic	Double precision
MILP solver arithmetic	Double precision
Absolute rank threshold τ_{abs}	0
Relative rank threshold τ_{rel}	$\max\{m_A, n_A\}\varepsilon_{\mathrm{mach}}(A)$
Zero-column threshold τ_0	10^{-12}
Column-comparison tolerance η_{abs}	10^{-13}
Column-comparison tolerance η_{rel}	10^{-10}
Duplicate grouping precision	12 decimal places
MILP residual bound ϵ_{eq}	10^{-8}
Compressed projection tolerance ϵ_{c}	$\max\{10^{-7}, \sqrt{d}\epsilon_{\mathrm{eq}}\}$
Full projection tolerance ϵ_{f}	10^{-7}
Random objective perturbation weight	0
Requested relative MIP gap	0
Maximum enumeration iterations per layer	1000
Time limit per MILP solve	300 s
Total enumeration time budget per layer	900 s

The exact coverage condition remains the rank-preservation condition of Theorem 1. Stage II constrains sequences constructed from the retained pools; it cannot recover a missing input pattern by candidate selection alone.

D Stateful Reconstruction and Pruning

D.1 Preservation of the true batch

Assume that every true spike pattern belongs to its corresponding candidate pool, the forward dynamics and initial states are exact, and all rank comparisons are exact. Let $\mathcal{J}$ index the constrained downstream representations and define $\mathcal{K} := \{0\} \cup \mathcal{J}$, where $j = 0$ denotes the input.

Consider replaying the true input batch in the fixed search order. At each position, the true input vector belongs to $\mathcal{P}_0$. Starting from the known initial states, deterministic propagation reproduces the true neuronal states and downstream spikes. By induction over timesteps, every downstream membership check therefore succeeds.

The occurrence constraints also hold. For each $j \in \mathcal{K}$, the gradient factorization gives

$$r_j \le \mathrm{rank}(\widetilde{S}^{(j)}) \le k_j^\star \le L,$$

where $k_j^\star$ is the number of distinct true spike patterns at representation j. If $r_j = L$, all L patterns are distinct, so the no-reuse bound is valid. Otherwise, the permitted maximum of L occurrences includes every possible true occurrence count.

The terminal rank checks hold because replaying the true batch reproduces $\widetilde{S}^{(j)}$, whose rank is at least r_j. In the implementation, the rank of a reconstructed matrix $\widehat{\widetilde{S}}^{(j)}(X)$ is computed exactly over $\mathbb{Q}$, equivalently over $\mathbb{R}$, using fraction-free elimination. These matrices are binary with an appended row of ones, so no numerical threshold is used for their ranks; numerical uncertainty enters through the gradient ranks r_j. For prefix pruning, suppose q positions have been assigned and $k_{j,q}$ distinct true patterns have appeared. Each remaining position introduces at most one additional pattern, so

$$r_j \le k_j^\star \le k_{j,q} + L - q.$$

Consequently, the pruning condition $k_{j,q} + L - q < r_j$ never holds along the true branch.

Thus, the true batch remains feasible and none of its prefixes is pruned. Because the candidate pools and search depth are finite, depth-first search without a computational cutoff returns a feasible batch

whenever one exists. If the constraints uniquely determine the multiset of complete input trajectories, the returned batch is exact, including repeated examples, irrespective of example ordering. Returning the first feasible batch does not itself establish uniqueness.

D.2 BACKTRACKING SEARCH

Algorithm 3 fills examples in a fixed order and their timesteps chronologically. Each example starts from the known initial state V_0. Occurrence counts are shared across the batch, whereas neuronal states are maintained separately.

Each recursive extension uses independent copies of the batch, states, and counts. Rejected branches therefore leave their parent unchanged, and backtracking can revise previously completed examples. The recursive routine shares the fixed inputs of SEARCH and a counter ν of attempted candidate extensions. The budget Q limits this counter; $Q = \infty$ disables the cutoff. The routine FORWARDSTEP returns the updated neuronal state and induced spikes at all constrained downstream representations. Unseen patterns have occurrence count zero.

Algorithm 3 Stateful consistency-guided search

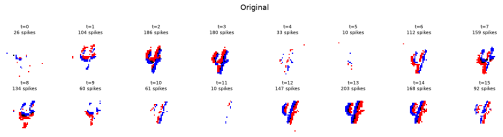
function SEARCH($\{\mathcal{P}_j\}_{j\in\mathcal{K}}, \theta, V_0, \{r_j\}_{j\in\mathcal{K}}, B, T, Q$)
  $L \leftarrow BT$, $\nu \leftarrow 0$
  $c_j \leftarrow 1$ if $r_j = L$, and $c_j \leftarrow L$ otherwise, for every $j \in \mathcal{K}$
  Initialize empty batch X and counts $n \leftarrow 0$
  $V_a \leftarrow$ COPY(V_0) for $a = 1, \ldots, B$
  return BACKTRACK($0, X, V, n$)
end function
function BACKTRACK(q, X, V, n)
  if $q = L$ **then**
    if $\operatorname{rank}(\widehat{\widetilde{S}}^{(j)}(X)) \geq r_j$ for every $j \in \mathcal{K}$ **then**
      return $(X,$ FEASIBLE$)$
    end if
    return $(\varnothing,$ EXHAUSTED$)$
  end if
  Let (a, t) be the next position in the fixed schedule
  for all $x \in \mathcal{P}_0$ **do**
    if $\nu \geq Q$ **then**
      return $(\varnothing,$ BUDGET$)$
    end if
    $\nu \leftarrow \nu + 1$
    $(X', V', n') \leftarrow$ COPY(X, V, n)
    $(V'_a, \{y_j\}_{j\in\mathcal{J}}) \leftarrow$ FORWARDSTEP(θ, V'_a, x)
    $y_0 \leftarrow x, \quad X'_{a,t} \leftarrow x$
    $n'_j(y_j) \leftarrow n_j(y_j) + 1$ for every $j \in \mathcal{K}$
    $k'_j \leftarrow |\{p : n'_j(p) > 0\}|$ for every $j \in \mathcal{K}$
    if $y_j \in \mathcal{P}_j$ for every $j \in \mathcal{J}$, $n'_j(y_j) \leq c_j$ and $k'_j + L - q - 1 \geq r_j$ for every $j \in \mathcal{K}$ **then**
      $(\widehat{X}, \sigma) \leftarrow$ BACKTRACK$(q + 1, X', V', n')$
      if $\sigma \neq$ EXHAUSTED **then**
        return $(\widehat{X}, \sigma)$
      end if
    end if
  end for
  return $(\varnothing,$ EXHAUSTED$)$
end function

A feasible return satisfies the imposed constraints but does not certify uniqueness. Exhaustion means that no feasible batch exists within the supplied pools under those constraints. Budget termination establishes neither feasibility nor exhaustion.

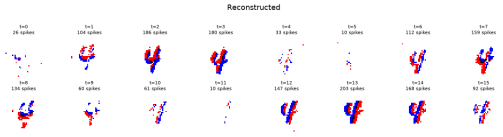
Search complexity and stopping. Let $\kappa = |\mathcal{P}_0|$. The unpruned search has depth $L = BT$ and at most $\sum_{q=0}^{L} \kappa^q$ prefix nodes. Thus, the worst case remains exponential, although membership checks, occurrence bounds, and rank pruning reduce the explored tree in practice. The implementa-

Table 4: Dataset-specific input representations and SNN architectures.

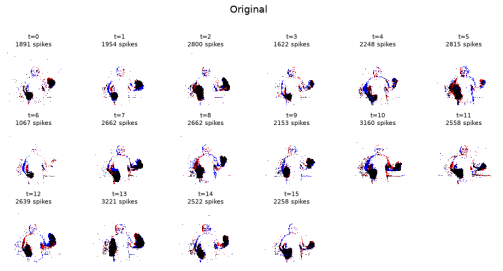
Dataset	Input	Architecture
N-MNIST	$34\times34\times2$	$2312\to512\to256\to128\to10$
DVS128 Gesture	$128\times128\times2$	$32768\to7384\to4096\to1024\to11$
N-Caltech101	$64\times64\times2$	$8192\to1024\to512\to256\to101$
CIFAR10-DVS	$128\times128\times2$	$32768\to7384\to4096\to1024\to10$



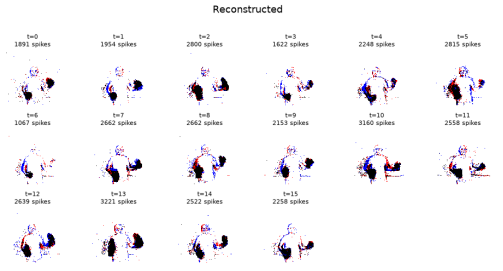
(a) N-MNIST, $B = 1, T = 16$, original



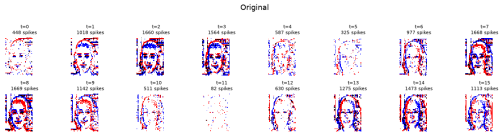
(b) N-MNIST, $B = 1, T = 16$, reconstructed



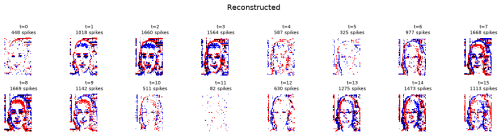
(c) DVS128 Gesture, $B = 1, T = 16$, original



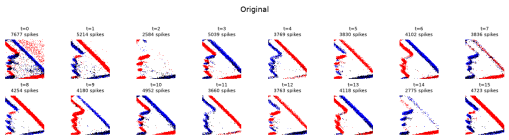
(d) DVS128 Gesture, $B = 1, T = 16$, reconstructed



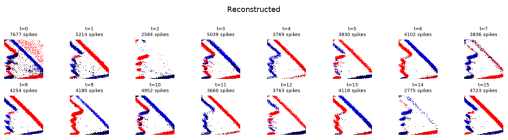
(e) N-Caltech101, $B = 1, T = 16$, original



(f) N-Caltech101, $B = 1, T = 16$, reconstructed



(g) CIFAR10-DVS, $B = 1, T = 16$, original



(h) CIFAR10-DVS, $B = 1, T = 16$, reconstructed

Figure 6: Single-example temporal reconstructions across datasets for $B = 1$ and $T = 16$. Each pair shows the original spike sequence and the reconstructed spike sequence for the same input.

tion returns the first feasible batch without certifying uniqueness. We distinguish this outcome from exhaustion of the search tree over the retained pools and termination at a computational limit.

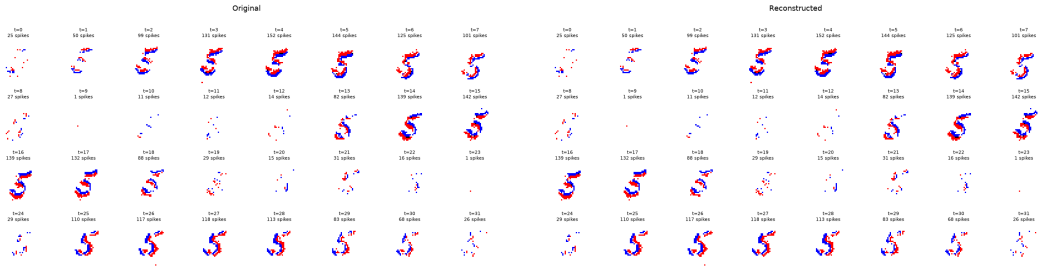
E Experiments

E.1 Architectures and Experimental Settings

E.2 Exact Reconstruction across Datasets

Single-example reconstruction across timesteps. Figure 6 reports representative single-example reconstructions for the four event-based vision datasets with $B = 1$ and $T = 16$. For each dataset, we show the original spike sequence and the corresponding reconstructed sequence over time. The reconstructions match the original spike frames, illustrating that the attack recovers the temporal spike sequence itself, not only its temporal aggregation.

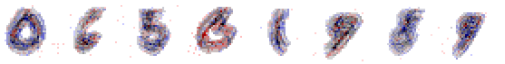
Figure 7 shows an additional N-MNIST example with a longer temporal horizon, $B = 1$ and $T = 32$. This example illustrates exact single-example recovery over a longer spike sequence.



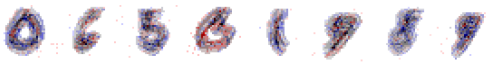
(a) N-MNIST, $B = 1, T = 32$, original (b) N-MNIST, $B = 1, T = 32$, reconstructed

Figure 7: Single-example temporal reconstruction on N-MNIST with $B = 1$ and $T = 32$.

Mini-batch reconstruction. Figure 8 reports representative mini-batch reconstructions for the four datasets with $B = 8$ and $T = 8$. Each image is aggregated over timesteps for visualization only. The reconstruction is exact up to permutation of the examples within the batch, since the gradient does not encode an intrinsic ordering of complete batch examples.



(a) N-MNIST, $T = 8, B = 8$, original



(b) N-MNIST, $T = 8, B = 8$, reconstructed



(c) DVS128 Gesture, $T = 8, B = 8$, original



(d) DVS128 Gesture, $T = 8, B = 8$, reconstructed



(e) N-Caltech101, $T = 8, B = 8$, original



(f) N-Caltech101, $T = 8, B = 8$, reconstructed



(g) CIFAR10-DVS, $T = 8, B = 8$, original



(h) CIFAR10-DVS, $T = 8, B = 8$, reconstructed

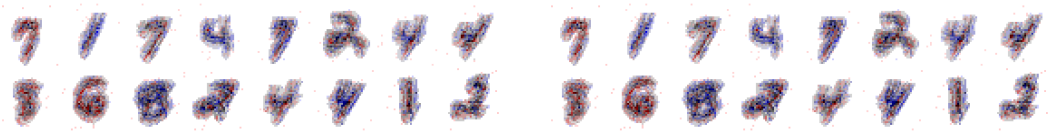
Figure 8: Mini-batch reconstructions across datasets for $T = 8$ and $B = 8$. Images are aggregated over timesteps for visualization. Reconstruction is exact up to permutation of complete examples within the batch.

Figure 9 shows additional N-MNIST mini-batch reconstructions for $T = 8$ and increasing batch sizes. The attack recovers all examples exactly up to batch permutation, including the largest shown setting with $B = 32$.

E.3 Sparsity and Candidate-Coverage Diagnostics

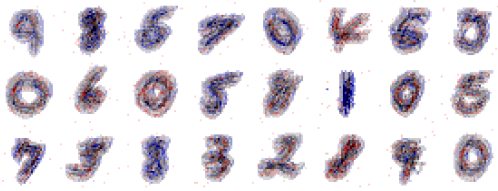
Table 5 reports the temporal-horizon diagnostics used in Section 3.3. As T increases, the input spike sequence becomes sparser and the candidate pool grows. Exact recovery succeeds up to $T = 32$. At $T = 36$, the true-pattern coverage remains complete, but Stage II does not find a complete reconstruction within the configured search budget. At $T = 48$, coverage is incomplete, so some true input patterns are missing from the retained candidate pool.

Table 6 reports the gradient-sparsity diagnostics for the tested fully connected settings on N-MNIST with $B = 1$ and $T = 8$. The table shows the sparsity of the first-layer gradient factor, the retained true-pattern coverage, and the reconstruction status. These results are used only to study how first-layer gradient sparsity affects the attack.

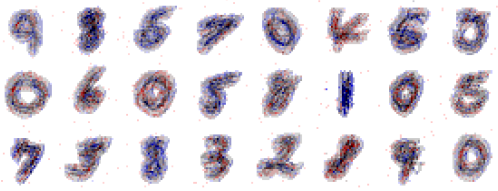


(a) N-MNIST, $T = 8$, $B = 16$, original

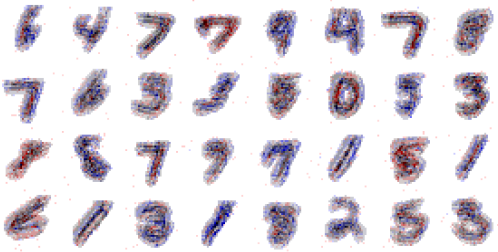
(b) N-MNIST, $T = 8$, $B = 16$, reconstructed



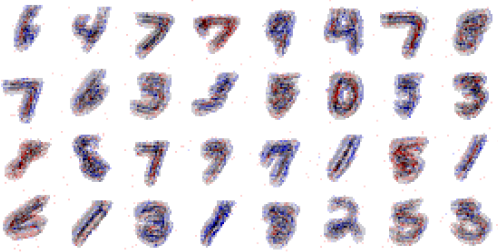
(c) N-MNIST, $T = 8$, $B = 24$, original



(d) N-MNIST, $T = 8$, $B = 24$, reconstructed



(e) N-MNIST, $T = 8$, $B = 32$, original



(f) N-MNIST, $T = 8$, $B = 32$, reconstructed

Figure 9: N-MNIST mini-batch reconstructions for $T = 8$ and increasing batch sizes. Images are aggregated over timesteps for visualization. Reconstruction is exact up to permutation of complete examples within the batch.

Table 5: Temporal-horizon diagnostics on N-MNIST with $B = 1$. Spike sparsity is reported as $1-$spike density.

T	Spike sparsity	Candidate pool	True coverage	Status
4	85.76%	4	100%	success
6	88.85%	6	100%	success
8	91.37%	8	100%	success
12	93.41%	12	100%	success
16	94.54%	16	100%	success
24	95.75%	24	100%	success
32	96.40%	60	100%	success
36	96.62%	496	100%	search exhausted
48	97.13%	163	2.08%	incomplete coverage

Table 6: First-layer gradient-sparsity diagnostics on N-MNIST with $B = 1$ and $T = 8$.

FC setting	First-layer G sparsity	True coverage	Status
2 layers	4.10%	100%	success
3 layers	26.73%	100%	success
4 layers	52.73%	100%	success
6 layers	93.84%	100%	success
7 layers	97.52%	100%	success
8 layers	92.84%	100%	success
9 layers	99.86%	50.00%	search exhausted
10 layers	100.00%	12.50%	search exhausted