# READING BETWEEN THE CLAUSES: DATA RECONSTRUCTION ATTACK ON FEDERATED TRAINING OF TSETLIN MACHINES

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Tsetlin Machines (TMs) classify binary inputs through learned logical combinations of features. They have recently gained attention because they train without GPUs, run efficiently on edge devices, and are explainable by design: each learned clause is a small pattern that points directly at the input pixels behind a decision. Weighted Convolutional TMs (wCTMs) have even been shown to outperform neural networks on federated image classification under strong data heterogeneity. Federated learning is meant to protect privacy by keeping each client's data on their own device and sharing only the trained model with the server.

We show that for wCTMs this protection fails: the same explainability that makes a clause readable also makes it invertible. Under heterogeneous data, many clients hold classes with so few images that their clauses memorize the individual samples. We present the first data reconstruction attack on federated wCTM training. From a single uploaded model, an honest-but-curious server estimates how many images a class was trained on and reassembles the clauses' learned patches to recreate the original images. No auxiliary data and no protocol manipulation are required. Across five benchmarks, the attack reconstructs hundreds to thousands of images, up to 95% of the images each heterogeneous setting leaves exposed.

## 1 INTRODUCTION

Tsetlin Machines (TMs) Granmo (2018) have recently attracted growing attention as a logic-based alternative to neural networks. They train without GPUs Abeyrathna et al. (2021b), map cleanly onto hardware and edge devices Maheshwari et al. (2023), and are explainable by design Abeyrathna et al. (2023). They have proven especially effective in heterogeneous federated learning, where they reach state-of-the-art accuracy and even surpass neural networks on some tasks How et al. (2025; 2023). At their core, TMs take binary data as input and represent knowledge as clauses: logical formulas, each a conjunction of input features required to be present or absent. Clauses are learned by reinforcement so that they capture patterns typical of a class. Because they are learned separately per class, a varying number of samples in one class affects only that class, which is what makes TMs robust under the non-independent and identically distributed (non-IID) data of federated learning. Among the many TM variants, weighted Convolutional TMs (wCTMs), combining clause weights Abeyrathna et al. (2021a) with convolutional clauses Granmo et al. (2019), deliver the strongest performance on federated image classification. Yet, their privacy implications, especially in federated learning, are little understood and have not been studied so far.

Federated learning is meant to protect privacy by keeping each client's data on their own device and sharing only the trained model with the server, which aggregates the client models into a global one. Prior work has shown that data reconstruction attacks are a serious threat. For neural networks they are well studied: a passive, honest-but-curious server can already invert client gradients (Zhu et al., 2019; Geiping et al., 2020), and an active server that manipulates the shared model recovers data even more readily (Fowl et al., 2022; Boenisch et al., 2023). Beyond neural networks, reconstruction has been shown for other model families, including interpretable models such as decision trees and random forests (Fredrikson et al., 2015; Ferry et al., 2024b) and, in the federated tree setting, gradient-boosted ensembles (Di Gennaro et al., 2025). None of these transfers to wCTMs. The neural attacks require a differentiable model with gradients to invert, which a wCTM has neither



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: A selection of high-quality reconstructions produced by our attack on five data sets. The first row shows original images and the second row the reconstructed binarized training samples. The originals are matched with the reconstructions after binarization by hamming distance.

of. Although TimberStrike (Di Gennaro et al., 2025) likewise attacks a non-differentiable federated model, it exploits the specific structure of boosted trees, real-valued split thresholds and per-node gradient statistics, that a booleanized, clause-based wCTM simply does not expose. Reconstructing training data from federated wCTMs therefore remains an open problem.

We show that with access to a single client's model, the server can reconstruct the preprocessed training images of a wCTMs class whenever that class was trained on only a handful of samples (Figure 1). Two properties make this possible. The first is the non-IID setting itself: realistic federated data is distributed unevenly, so many clients hold classes supported by few samples. For instance, at non-IID settings with Dirichlet parameter $\alpha = 0.3$, nearly every client exposes at least one attackable class. The second is explainability, because a TM stores clauses that correspond directly to patterns in the training data. Training wants these patterns to generalize, but under strong heterogeneity they often do not, and clauses instead retain parts of the individual images.

Our attack turns this channel into a reconstruction pipeline. As wCTMs carry a separate set of clauses for each class, classes of a client can be attacked individually, which we denote as client classes. From the received models, the server first identifies the client classes that were trained on few enough samples to be vulnerable. For each such class our attack estimates the number of training images by exploiting a graph-theoretic property of the clauses, then solves an integer program that separates the clauses into overlapping sets. We prove and validate empirically that this estimation is a lower bound on the number of training samples. Each set is finally painted onto an empty canvas from the features its clauses encode, and the confident common pixels of many-sample client classes of the same class fill in the shared structure the clauses omit. Although some federated TM algorithms are iterative, a single round suffices for reconstruction without any prior on the data distribution and without altering the protocol. We evaluate the attack on five image-recognition benchmarks, with the data spread across many clients. The reconstructions are specific to individual samples rather than to classes: against class prototypes and real images of the target class, our attack is best on every metric, and it reconstructs up to $95\%$ of the exposed images. In total, depending on the data set, the attack recovers hundreds to thousands of training images across the federation.

**Contributions & Structure** We (i) present the first training data reconstruction attack against federated wCTMs under data heterogeneity by utilizing the internal structure of wCTMs and exploiting their inherent explainability; (ii) characterize why and when federated wCTM training is vulnerable and explore potential countermeasures; and (iii) evaluate the attack on five image-recognition benchmarks in realistic federated settings, together with an extensive ablation study.

Section 2 introduces the necessary background, our threat model, and the notation. Section 3 develops the attack, from intuition to the details of each phase. Section 4 reports the implementation, experimental setup, and results. Section 5 discusses countermeasures and limitations of the attack. Section 6 surveys related work and Section 7 concludes.

## 2 Preliminaries

In this section, we give a short overview of the most important concepts for this work, namely weighted Convolutional Tsetlin Machines and their application in federated learning. In the end, we state our threat model. For more details on training, see Appendix B.

### 2.1 Weighted Convolutional Tsetlin Machine

We attack weighted Convolutional Tsetlin Machines (wCTMs), which combine the convolutional clauses of Granmo et al. (2019) with the integer clause weights of Abeyrathna et al. (2021a), and are designed for classification tasks. An input is a binarized image $\mathbf{x} \in \{0,1\}^{R \times R \times B}$ of resolution $R$ with $B$ channels, obtained by thresholding the original pixels. We index its pixels by $v \in \mathcal{V} := [R] \times [R] \times [B]$, write $x_v$ for the value at pixel $v$, and abbreviate $[N] := \{1, \ldots, N\}$. Each image is to be assigned one of $L$ labels $l \in [L]$. A wCTM classifies through a collection of clauses, each of which is a small conjunctive pattern over a $p \times p$ patch that may be matched anywhere in the image. We describe a clause $C_i$ by the two disjoint sets of patch coordinates it constrains, $\mathcal{O}_i,\ \mathcal{Z}_i \subseteq [p^2 B]$ with $\mathcal{O}_i \cap \mathcal{Z}_i = \emptyset$, listing the pixels the clause requires to be on and off, respectively, and by a learned set of valid positions $\mathcal{P}_i \subseteq [(R-p+1)^2]$ at which this pattern may be evaluated. $\mathcal{P}_i$ is usually restricted to a rectangular region but for simplicity we treat it as a set of pixel indices. Placing the pattern at a position $\pi \in \mathcal{P}_i$ translates the two coordinate sets onto the image, yielding the covered pixels $\mathcal{O}_{i,\pi},\ \mathcal{Z}_{i,\pi} \subseteq \mathcal{V}$, and the clause fires whenever its pattern matches at some position,

$$C_i(\mathbf{x}) = 1 \iff \exists\, \pi \in \mathcal{P}_i :\ x_v = 1\ \ \forall v \in \mathcal{O}_{i,\pi}\ \text{ and }\ x_v = 0\ \ \forall v \in \mathcal{Z}_{i,\pi}. \tag{1}$$

For a class $l$, half of its clauses form a positive bank $\mathcal{C}_l^+$ that votes for the class and half a negative bank $\mathcal{C}_l^-$ that votes against it, and the prediction is the label whose weighted vote is largest,

$$\hat{y} \;=\; \arg\max_{l \in [L]} \left( \sum_{C_i \in \mathcal{C}_l^+} w_i\, C_i(\mathbf{x}) \;-\; \sum_{C_i \in \mathcal{C}_l^-} w_i\, C_i(\mathbf{x}) \right). \tag{2}$$

Every clause additionally carries a learned integer weight $w_i \in \mathbb{Z}_{>0}$ that scales its vote, letting one clause stand in for several identical copies of itself and thus keeping the model compact. The total number of clauses per class is denoted by $T$. Each candidate literal $j$ of a clause $C_i$, a patch pixel in its on- or off-form, per channel, is controlled by a Tsetlin automaton with an integer state $a_{i,j} \in [-U, U]$ called TA state, and the literal enters the clause when $a_{i,j} > 0$, so that $\mathcal{O}_i$ and $\mathcal{Z}_i$ are simply read off the automata that have crossed zero. Training streams labelled images for $e$ epochs past the machine one by one and, through reinforcement-style feedback governed by a specificity hyperparameter $s$ and a target margin $Y$, moves the TA states and the weights so that clauses come to capture patterns typical of their class. The precise feedback rules are given in Appendix B. As the literals of a clause directly correspond to pixels of the input images, clauses can be visualized naturally as shown in Figure 2.

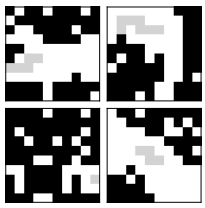


Figure 2: Clause visualization: black/gray correspond to off/on literals while white is neutral.

### 2.2 Problem Setting and Threat Model

We consider the standard cross-silo federated setup of $n$ clients coordinated by a single central server. A data set $\mathcal{D} = D_1 \cup \cdots \cup D_n$ is partitioned across the clients, with client $i$ holding the subset $D_i$. This partition is not uniform but follows the non-IID distributions typical of realistic deployments: a given class may be represented on one client by only a few images and on another by a large number. We focus on image classification, which has been the primary target of federated Tsetlin Machine training How et al. (2023; 2025); Gohari et al. (2025). Thus, we use the terms sample and image interchangeably. Each client $i$ holds its own wCTM and trains it locally on $D_i$. Once local training completes, the server receives the $n$ trained wCTMs. The local model configuration and the corresponding hyperparameters are publicly known. The server is an honest-but-curious adversary, and they seek to infer as much as possible about the clients' private data. They neither collude with any client nor manipulate the training in any way. Their only leverage is white-box access to the received models, which the protocol grants by construction, since aggregation requires the model internals. In particular, the attack requires no auxiliary data and no prior knowledge of the clients' data distributions. The server attacks each class $l$ of each client's $i$ wCTM separately and thus a single client class is a pair $(i, l)$.

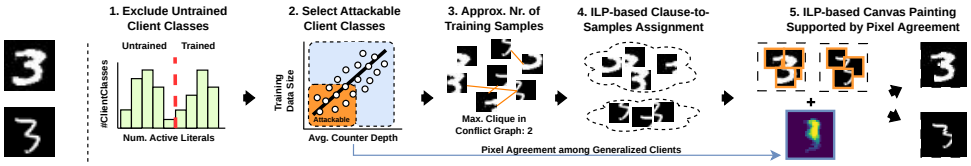



Figure 3: Overview of our reconstruction algorithm illustrated based on class three of MNIST. 1.), 2.) The server has to filter out non-attackable client classes, i.e. classes trained on no data at all or trained on many samples. 3.) The server determines a lower bound on the number of training samples based on the maximum number of pairwise incompatible clauses. 4.) An ILP solver is used to assign clauses to their respective samples. 5.) A second ILP places the clauses on each canvas, with high-agreement pixels from many-sample client classes providing additional support.

# 3 Attack Method

## 3.1 Intuition

In state-of-the-art federated Tsetlin Machine training clients have to upload their locally trained wCTMs directly to the server, which combines the models into a global model to be redistributed to the clients How et al. (2025; 2023). Two properties of the setting make reconstruction possible: the inherent explainability of wCTMs, and the heterogeneity of the federated non-IID regime. The explainability of Tsetlin models stems from the transparent relationship between an activated clause and the pixels responsible for its activation: a clause is a small, directly readable pattern, so its literals point straight back at the input. This is a virtue when the goal is to understand a prediction, but it becomes a liability the moment a clause encodes a real training image rather than a general class pattern. Federated heterogeneity leaves some clients with only a handful of images in a given class, and wCTM training, which is meant to distill patterns that generalize to unseen data, instead begins to remember parts of those few images when they are all it has to learn from, as Figure 4 illustrates.

Putting the two together, the server holds white-box models in which a substantial fraction of client classes was trained on only a few samples, so that the interpretability of each clause becomes a direct window onto small parts of the underlying training images. Because a Tsetlin model exposes each class independently, we can instead target one client class at a time, and we accordingly omit the label index $l$ throughout this section. However, the server does not know how to put clauses of a single client class together to reconstruct training samples. Informally, reconstruction now resembles reassembling several jigsaw puzzles whose pieces, the clauses, have all been tipped into a single pile: pieces overlap heavily, some are duplicated, some belong to no pile at all. Despite this, we show that the server can recover a lot of training images across the participating clients.

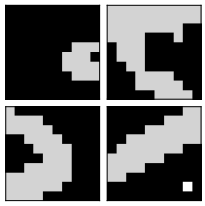


Figure 4: Overfitted clauses trained on class 5 of MNIST.

The attack identifies client classes trained on few enough samples to have memorized, yet not so few as to carry no signal. Then, it estimates the number of training samples behind such a class, assigns the class's clauses to these putative samples, and finally reconstructs each image from its assigned clause set. Figure 3 gives an overview of the attack and the pseudocode can be found in Appendix C.

## 3.2 Estimating the Number of Training Samples

We assume for now that a vulnerable client class has been identified. We restrict attention to its positive clauses $\mathcal{C}^+$, the ones trained to recognize the class, and refer to these simply as clauses for the remainder of the attack. We first discard two kinds of uninformative clauses. Clauses with few literals, with $|\mathcal{O}_i| + |\mathcal{Z}_i|$ small, constrain too little of the image to aid reconstruction and only add noise, and clauses whose weight $w_i$ falls below a small threshold are unlikely ever to have mattered during training and thus carry no learned pattern. Reconstruction requires knowing how many images $M$ the targeted class was trained on, yet a wCTM records this nowhere. However, in the low-sample regime, wCTMs tend to partition their clauses into near-disjoint subsets, each

responsible for one training image, rather than sharing clauses across images. Because these subsets are never meant to fire together, the model has no reason to keep clauses from different subsets mutually consistent, and clauses belonging to different images therefore tend to conflict. We call two clauses conflicting when no input can activate both at once, or put differently, when the conjunction of their patterns, across every admissible pair of positions, is contradictory. The largest number of pairwise conflicting clauses is then an estimate of the number of training samples.

Enumerating conflicting subsets directly is infeasible: even simple data sets use at least 100 clauses per class, giving millions of combinations already for $M \geq 4$. Thus, we formulate the problem as a graph $G = (\mathcal{C}^+, \mathcal{K})$ whose vertices are the positive clauses and whose edges join conflicting pairs,

$$\mathcal{K} \;=\; \Big\{(C_i, C_j) \in \mathcal{C}^+ \times \mathcal{C}^+ \text{ for } i < j \;\Big|\; \nexists\, \mathbf{x} \in \{0,1\}^{R\times R\times B} :\; C_i(\mathbf{x}) = C_j(\mathbf{x}) = 1\Big\}.$$

A set of pairwise conflicting clauses is a clique in $G$, so the largest such set is the maximum clique $Q$ of size $\omega(G)$, and we take its size $\hat{M} = |Q|$ as our estimate. The estimate is a lower bound, $\hat{M} \leq M$: any clauses forming a clique pairwise conflict, so each must be supported by a different training image, which injects the clique into the $M$ samples. Here, the weight filter approximately enforces that every remaining clause was activated by at least one training image. We make this precise, and characterize when the bound is tight, in Figure 12 and Appendix D. As the evaluation shows, $\hat{M}$ is accurate enough in the low-sample regime to anchor the reconstruction that follows.

### 3.3 Preparing Canvases

The estimated number of training samples $\hat{M}$, based on the size of the maximum clique, tells us how many piles of puzzle pieces there could have been, but not yet which pieces belong to which pile. The next step is therefore to obtain a partition of clauses by deciding which positive clauses carry information about which training image. We create $\hat{M}$ empty placeholder images, called canvases, where each pixel can be set either to 0 or to 1, and seek, for each canvas $k \in [\hat{M}]$, a set of positive clauses $\mathcal{C}_k^+ \subseteq \mathcal{C}^+$ that can later be painted onto it to reconstruct one training image. Note that these sets do not have to be pairwise disjoint as learned patterns might be useful for multiple samples.

Three observations guide the assignment. First, the maximum clique $Q = \{C_{q_1}, \ldots, C_{q_{\hat{M}}}\}$ of the previous phase supplies one clause per canvas that is guaranteed to belong to a distinct image: we call these clauses anchors and seed canvas $k$ with its anchor $C_{q_k}$. Since the anchors are pairwise conflicting, no two anchors can share a canvas. The pairing of anchors to canvases is otherwise arbitrary, as the canvases are interchangeable at this stage. Second, any remaining clause was reinforced by at least one training image and should appear somewhere, so every retained clause is placed on at least one canvas. Third, and hardest, each set $\mathcal{C}_k^+$ must be internally conflict-free, since clauses that conflict cannot all be painted onto the same canvas. We cast these requirements as the integer linear program in Figure 5, where the binary variable $z_{i,k}$ is 1 exactly when $C_i \in \mathcal{C}_k^+$. Its objective maximizes the weight of the placed clauses, encouraging the solver to commit as much ink to the canvases as possible, while the constraints pin each anchor to its canvas, place every clause at least once, and forbid conflicting pairs from sharing a canvas. Solving it yields the clause sets $\mathcal{C}_1^+, \ldots, \mathcal{C}_{\hat{M}}^+$.

$$\begin{aligned}
&\max_{z} \sum_{k=1}^{\hat{M}} \sum_{C_i \in \mathcal{C}^+} w_i\, z_{i,k} \quad \text{s.t.} \\
&z_{q_k,k} = 1 && \forall\, k \in [\hat{M}], \\
&\sum_{k=1}^{\hat{M}} z_{i,k} \geq 1 && \forall\, C_i \in \mathcal{C}^+, \\
&z_{i,k} + z_{j,k} \leq 1 && \forall\, (C_i, C_j) \in \mathcal{K}, \\
& && \quad k \in [\hat{M}], \\
&z_{i,k} \in \{0,1\} && \forall\, C_i \in \mathcal{C}^+,\, k \in [\hat{M}].
\end{aligned}$$

Figure 5: ILP distributing the positive clauses $\mathcal{C}^+$ over the $\hat{M}$ canvases.

Pairwise conflict-freeness is necessary but not sufficient: three or more clauses can be pairwise compatible yet impossible to satisfy together, introducing a higher-order conflict. Enumerating all such conflicts as constraints is infeasible, as there may be exponentially many, so we handle them lazily. We solve the ILP, then check each resulting canvas with a SAT solver to test whether all its clauses can be activated by a common image. This is fast since clauses are simply propositional formulas. If some canvas fails, we exclude that assignment with an additional constraint and resolve. Since each round rules out at least one infeasible assignment, the procedure terminates, and in practice it returns a valid solution after only a few iterations.

### 3.4 Painting Each Canvas

Now, we can start painting each canvas separately as we are given a set of clauses each. The clause assignment of the previous phase guarantees that a conflict-free placement of every clause on its canvas exists, but it does not tell us where each clause sits exactly. We try to recover the positions with a second integer linear program by using one per canvas $k \in [\hat{M}]$. The ILP is shown in Figure 6.

We regard a canvas as a pixel grid whose cells are indexed by $v$ and carry values $x_v \in \{0, 1\}$. A clause may be placed only at a position from which its patch stays within bounds, so we introduce a binary variable $z_{i,\pi}$ that is 1 when clause $C_i$ is placed at an admissible position $\pi \in \mathcal{P}_i$. A placement determines which pixels the clause fixes: recall that $\mathcal{O}_{i,\pi}$ and $\mathcal{Z}_{i,\pi}$ are the canvas pixels the clause forces on and off when anchored at $\pi$. The ILP maximizes the weight of the placed clauses, to commit as many as possible. Its constraints place each canvas's anchor $C_{q_k}$ exactly once, place every other clause at most once and enforce consistency between placements and pixels. Whenever a clause is placed, the constraints drive its on-pixels to 1 and its off-pixels to 0, so that no two placed clauses ever demand opposite values for the same pixel. Reading off the $x_v$ as a grid yields the painted canvas by treating unconstrained $x_v$ as off.

$$
\begin{aligned}
&\max_{z,x} \sum_{C_i \in \mathcal{C}_k^+} w_i \sum_{\pi \in \mathcal{P}_i} z_{i,\pi} \quad \text{s.t.} \\
&\sum_{\pi \in \mathcal{P}_{q_k}} z_{q_k,\pi} = 1, \\
&\sum_{\pi \in \mathcal{P}_i} z_{i,\pi} \le 1 && \forall\, C_i \in \mathcal{C}_k^+, \\
&x_v \ge z_{i,\pi} && \forall\, (i,\pi),\ v \in \mathcal{O}_{i,\pi}, \\
&x_v \le 1 - z_{i,\pi} && \forall\, (i,\pi),\ v \in \mathcal{Z}_{i,\pi}, \\
&z_{i,\pi},\ x_v \in \{0,1\}.
\end{aligned}
$$

Figure 6: ILP arranging the clauses $\mathcal{C}_k^+$ conflict-free on canvas $k$.

Using positive clauses alone leaves a canvas incomplete. A clause remembers the pixels that make its class distinguishable, and this omits pixels that are common to the class, or shared across classes. To recover them we exploit that the server holds every client's model, and in particular the many-sample client classes. We take the client classes which have been trained on the most samples, by ranking models by their TA state depth, and with them, we build a pixel agreement map (Figure 7): each model votes for a pixel whenever one of its clauses wants that pixel activated. Then, we separate the foreground and background based on the highest between-class variance. These pixels are then switched on, excluding any fixed by the placement ILP or any constrained by a clause of the targeted client class.

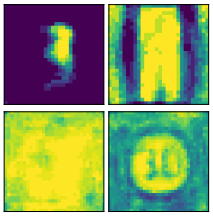


Figure 7: Pixel agreement maps for different data sets.

### 3.5 Detecting Vulnerable Client Classes

The one piece still missing making our attack practical is how the server decides which client class it can attack with a reasonable reconstruction quality. As our evaluation shows in more detail later (Figure 12), the sample estimation is almost exact for $M \le 5$ and increasingly underestimates beyond, which motivates $M \le 5$ as the attackable regime of our attack. As a correct number of canvases forms the basis of our attack, we introduce two independent tests to distinguish attackable from non-attackable client classes.

First, we have to rule out client classes that were never trained on any sample ($M = 0$). The server first checks whether the clause weights $w_i$ are still at their initial value of one. This flag alone is imperfect, as a class trained on a single sample can leave the weights nearly untouched as well. Interestingly, the numbers of active literals per clause form a nearly bimodal distribution which is what distinguishes a one-sample class from an untrained one and rescues it from being discarded.

At the other end, the server must avoid client classes trained on many samples ($M \ge 6$), whose wCTMs no longer support sample-specific reconstructions. A clause of a client class trained on multiple samples received a lot of feedback, so its TA states $a_{i,j}$ are pushed close to the extremes $\pm U$. A clause reinforced by only a handful of samples leaves them less saturated. We exploit that the saturation level depends mainly on the model configuration rather than on the data: the server trains shadow models of the same, publicly known configuration on arbitrary dummy data unrelated to any client's, reads off the TA states magnitudes a fully trained model reaches, and uses them as a threshold above which a client class is excluded. We detail this procedure in Appendix E.

Table 1: Main results of our reconstruction attack on all data sets.

| | MNIST | F-MNIST | SVHN | GTSRB | CIFAR-10 |
|---|---|---|---|---|---|
| Detection (F1 ↑) | 0.95 | 0.95 | 0.92 | 0.96 | 1.00 |
| Sample Est. (Acc ↑) | 86% | 95% | 99% | 91% | 99% |
| #Recons. (% of $M \leq 5$ ↑) | 94% | 95% | 68% | 88% | 94% |
| Recon. (Jaccard Index ↑) | 0.80±0.22 | 0.81±0.23 | 0.82±0.21 | 0.71±0.27 | 0.81±0.18 |
| Recon. (HD ↓) | 27±31 | 74±64 | 457±287 | 527±286 | 518±246 |
| Recon. (SSIM ↑) | 0.75±0.16 | 0.39±0.13 | 0.21±0.11 | 0.24±0.16 | 0.27±0.10 |
| Matching (F1 ↑) | 0.91 | 0.93 | 0.93 | 0.92 | 0.99 |

# 4 Evaluation

## 4.1 Setup

We evaluate the attack on five image-classification benchmarks: MNIST, F-MNIST, SVHN, GTSRB, and CIFAR-10. The latter three are multichannel and all data sets except GTSRB have $10$ classes, while GTSRB has $43$. The training set is distributed over $100$ clients in a non-IID fashion, with the number of samples per client class drawn from a Dirichlet distribution with parameter $\alpha = 0.3$. Each client trains the same wCTM on its private subset for $e = 30$ epochs. We follow the original FedTMOS configuration How et al. (2025). Full preprocessing and configuration details are given in Appendix A. This configuration is used throughout unless stated otherwise.

To evaluate our attack, we use a F1 score to measure how well the attack detects attackable models, and we report the accuracy of the sample-count estimation. Then, we report the ratio of reconstructed samples to the total number of attackable samples. Reconstruction quality is assessed three ways: the Hamming distance (HD) between a reconstruction and the wCTM input, the image quality score SSIM between a reconstruction and the original image before binarization, and a set similarity metric between the clauses activated by the reconstruction and those activated by its assigned original, called the Jaccard index. Finally, the Matching F1 score computes whether the attack found the right images and the right number of them: each reconstruction is paired with the image it most resembles within the set of the client's training images in terms of HD, and counts as a hit if that image really belongs to the class under attack and no other reconstruction already claimed it. A perfect score means the attack produced exactly one reconstruction per training sample, with none missed or duplicated. The reconstruction metrics are measured per reconstructed image, which yields a standard deviation, while the rest are single ratios computed over all client classes at once. To separate sample-specific from class-generic reconstruction, we additionally compare against three baselines that use no information from the victim's model: images generated uniformly at random, a class prototype given by the pixel agreement map alone, and $M$ real test images of the target class. All baselines are evaluated on the same client classes the attack selects. Whenever a plot shows results for multiple data sets, we divide the HD by the total number of bits for each data set to preserve readability.

## 4.2 Main Results

Table 1 reports our main results. The attack first reliably decides which client classes are worth attacking: the detection F1 ranges from $0.92$ to $1.00$ across the five datasets, so untrained and many-sample client classes are filtered out with few mistakes. Sample-count estimation is accurate across all data sets ($86$–$99\%$), as our exclusion tests reliably separate non-attackable from attackable client classes and the clique-based estimator closely tracks the true count in the low sample regime. The estimate is exact for $M \leq 5$ and beyond that underestimates but never overestimates $M$ (Figure 12). On this basis, the attack reconstructs the large majority of the vulnerable samples ($M \leq 5$): $94$–$95\%$ on MNIST, F-MNIST, and CIFAR-10, $88\%$ on GTSRB, and $68\%$ on SVHN, the hardest case. The reconstructions are faithful rather than generic. The clause-assignment Jaccard index is high throughout ($0.71$–$0.82$), confirming that the recovered clause sets closely match those of the true training samples. Reconstruction quality is highest for MNIST and F-MNIST, whose single channel preserves the most structure under binarization. The three multi-channel datasets exhibit larger Hamming distances, but this is expected: their inputs are $3072$ bits against $784$ for the grayscale sets,

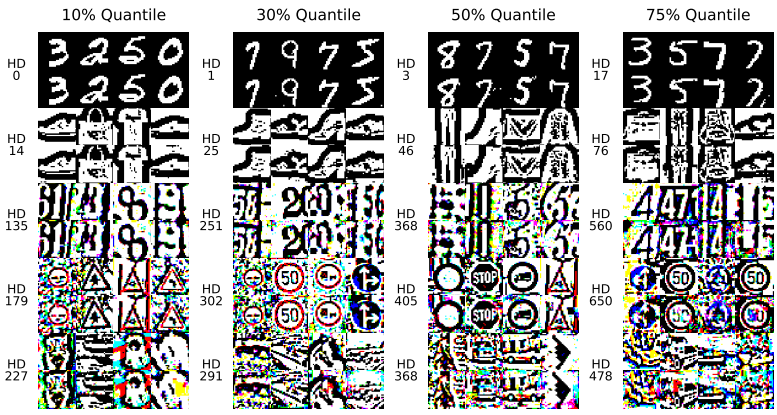



Figure 8: Reconstructions for multiple quantiles of Hamming distances. For each data set, the first row shows the inputs to the wCTMs after binarization and the second row the reconstructions.

so HD is meaningful within a dataset while SSIM gives the fairer cross-dataset comparison. Table 3 shows that our attack outperforms every baseline on every metric and data set. Perceptual metrics, however, are dominated by class structure: without agreement maps, our SSIM on F-MNIST and SVHN lies within $0.01$ of the class prototype. Matching F1 separates the two. Class prototypes and real class members match at most $30\%$ of the training samples under Matching F1, and as little as 2–4$\%$ on SVHN and CIFAR-10, while our attack matches 89–99$\%$ with or without agreement maps. The maps add class-common pixels and mainly lower the Hamming distance. Which image is recovered is determined by the victim's clauses. To give a sense of what the Hamming distance means visually, Figure 8 shows reconstructions across a range of HD quantiles: even above the 50th percentile the images remain clearly recognizable, losing only fine detail. A more fine-grained version is given in Figure 11.

### 4.3 Ablations

We now present the result of our ablation with the figures in Appendix G. Reconstruction quality is governed by how well the recovered clause set matches the true one: Hamming distance and Jaccard index are strongly correlated, and the bulk of reconstructions sit at high Jaccard with low HD, far from the random baseline (Figure 13). Across non-IID regimes the attack recovers almost all vulnerable samples ($M \leq 5$) and at least one sample from nearly every client (Figures 14 and 15), closely tracking the simulated ceiling of what is exposed. SVHN is the one consistent exception: its class imbalance leaves fewer attackable client classes and makes their detection harder, which is also why only $68\%$ of its vulnerable samples are recovered against 88–95$\%$ elsewhere. Generally speaking, the Dirichlet parameter $\alpha$ drastically influences how many client classes are vulnerable. Breaking the metrics down by the true sample count $M \in [7]$ (Figure 16) shows that both client class detection and sample estimation are reliable for $M \leq 5$ and degrade beyond it, so that a few classes with $M \geq 6$ are falsely attacked. Reconstruction quality likewise falls as $M$ grows, with a single exception at the step from one to two samples: a lone sample barely moves a clause's weights and literals, leaving too faint a footprint to invert.

To show the privacy-utility trade-off we train a global model with FedTMOS How et al. (2025) and report its accuracy (Figure 17). Note that our numbers deviate from the original, because it uses only 10 clients. More clients raise the number of vulnerable client classes, and the proportion of reconstructions remains consistent, while lowering FedTMOS accuracy by multiple percentage points. More local epochs sharply improve reconstruction quality, as clauses gain time to memorize details. Adding clauses lets a wCTM learn more patterns and improves the number of reconstructions, their quality, and FedTMOS accuracy. We also evaluate the SVHN setting with $T = 1100$ to match prior work, and it can be seen that our attack works especially well here. Raising the specificity $s$ has the opposite effect, degrading reconstruction remarkably, and a larger patch size $p$ helps the attack by enabling larger learned patterns while a very small patch size costs utility (Figure 18). Evaluating the runtime shows that it stays within a few seconds for the majority of client classes (Table 4).

## 5 Discussion

Our experimental results show that our attack can reconstruct samples from nearly every client, with the non-IID settings from prior work. Still, our attack recovers the preprocessed, binarized inputs a wCTM is trained on, which is a natural upper bound on any reconstruction attack on wCTMs. When a class is trained on more samples, its clauses start to generalize, our estimate degrades beyond $M \approx 5$ and reconstruction quality falls with it. This is a property of the model rather than of the estimator: once clauses generalize across images, they no longer conflict, and the model itself stops separating individual samples. Extending the non-IID setting to class-exclusion partitioning can reduce the number of attackable client classes by concentrating each class on fewer clients. Our evaluation targets one-shot and first-round uploads as in How et al. (2025). E.g., in later rounds in How et al. (2023), clients start from the global model, and the effect on the attack is future work.

Two natural defenses do not help. Secure aggregation Bonawitz et al. (2017) hides individual terms behind a sum, but the federated wCTM algorithms do not aggregate additively over models in the clear. FedTMOS reassembles a global model by rescaling, $k$-means clustering, and greedy reassignment How et al. (2025) and TPFL clusters clients by confidence Gohari et al. (2025), both needing per-client artifacts. Even where aggregation applies, as in FedTM How et al. (2023), a class held by a single client equals its own contribution, so nothing is concealed. Excluding low-sample clients likewise costs accuracy and discards the long tail needed for generalization Feldman (2020); Feldman & Zhang (2020). The principled defense is differential privacy: since the transmitted state is integer-valued, discrete mechanisms such as the distributed discrete Gaussian Kairouz et al. (2021) would add noise to the Tsetlin automaton states and clause weights and blur the literals the server exploits, though at a utility cost. The mitigation with immediate effect is the model configuration itself (Figures 17 and 18): minimizing the number of clauses, local epochs, and patch size subject to a utility floor, and raising specificity as far as accuracy allows, each trading utility for privacy.

## 6 Related Work

FedTM (How et al., 2023), the one-shot FedTMOS (How et al., 2025), which outperforms one-shot neural-network baselines, and the personalized TPFL (Gohari et al., 2025). All of them have clients upload their trained TMs to the server, which is what our attack exploits. Gradient inversion recovers training inputs from shared neural-network updates (Zhu et al., 2019; Zhao et al., 2020; Geiping et al., 2020; Jeon et al., 2021; Dimitrov et al., 2024; Petrov et al., 2024). Active variants use a malicious client (Hitaj et al., 2017) or a server that modifies the model to isolate samples (Fowl et al., 2022; Boenisch et al., 2023). Reconstruction from trained parameters alone has been shown for neural networks (Haim et al., 2022), and related leakage has been studied for model inversion (Fredrikson et al., 2015; Wu et al., 2024), embeddings (Song & Raghunathan, 2020) and diffusion models (Carlini et al., 2023). All of these rely on gradients of a differentiable loss or continuous parameters. A wCTM has integer states and no loss. Closest to our work are attacks on non-differentiable models. Ferry et al. (2024a) quantify leakage of interpretable models, and Ferry et al. (2024b) reconstruct the binary training set of a random forest also with constraint programming, but assume the number of training examples is known and do not handle spatially shared patterns. TimberStrike (Di Gennaro et al., 2025) lets a malicious client reconstruct tabular data from the split thresholds and decision paths of federated boosted trees. Our attack instead estimates the unknown sample count, places convolutional clauses that may match anywhere in the image, and works from a passive server's view. Appendix F gives details for the interested reader.

## 7 Conclusion

We presented the first data reconstruction attack on federated Tsetlin Machine training, targeting weighted Convolutional Tsetlin Machines in the non-IID regime for which they are promoted. It exploits their explainability, which maps clauses directly back to inputs, together with the many client classes that heterogeneity leaves trained on only a few samples. So the very interpretability that makes Tsetlin Machines attractive is what exposes their training data. From a single uploaded model, without auxiliary data or any deviation from the protocol, an honest-but-curious server reconstructs hundreds of training images across a federation of just $100$ clients, recovering up to $95\%$ of everything the non-IID setting leaves exposed.

## AI USE STATEMENT

We used generative AI tools as a general-purpose assistant throughout the preparation of this paper. The LLM assisted in four ways. For writing, it drafted and polished prose from author-provided bullet points and outlines. Every passage was reviewed, edited, and approved by the authors. For retrieval and discovery, it helped surface potential misses in related work, but every cited reference was located and verified against its primary source by the authors, who discarded suggestions they could not confirm. For the mathematical claims, it helped draft and stress-test the definitions and proofs. Any code produced with the assistance of generative AI was critically reviewed by the authors. The authors take full responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

## ETHICS STATEMENT

This work presents an attack that reconstructs private training data from federated Tsetlin Machine models, and we have considered its ethical implications carefully. All experiments use only public, standard benchmark datasets (Table 2). We reconstruct no real individuals' private data and conduct all attacks against models we trained ourselves in a controlled setting. The attack assumes an honest-but-curious server within the standard federated threat model and does not target any deployed system. In line with best practice for privacy-attack research, we accompany the attack with an analysis of countermeasures (Section 5). On balance, we judge that publishing this attack, together with its limitations and mitigations, strengthens the privacy of federated Tsetlin Machine learning rather than undermining it.

## REPRODUCIBILITY STATEMENT

We have taken several steps to make our results reproducible. The threat model and the full attack are specified in Sections 2.2 and 3, with the sample count estimator, the two integer programs, and the vulnerable class detection given as explicit formulations. The full pseudocode can be found in Appendix C. All datasets are public and listed with their splits in Table 2, and the exact preprocessing, federated partitioning, model configuration, solver choices and time budgets, attack hyperparameters, and hardware are reported in Section 4.1 and Appendix A. The data-free calibration of the detection gate, including the shadow-model construction and the resulting thresholds, is described in Appendix E. We build on the public FedTMOS and Tsetlin Machine implementations cited in Appendix A, and the code will be released upon acceptance to reproduce all reported figures and tables.

## REFERENCES


K Darshana Abeyrathna, Ole-Christoffer Granmo, Lei Jiao, and Morten Goodwin. The regression tsetlin machine: A tsetlin machine for continuous output problems. In *EPIA Conference on Artificial Intelligence*, pp. 268–280. Springer, 2019.

K Darshana Abeyrathna, Ole-Christoffer Granmo, Rishad Shafik, Alex Yakovlev, Adrian Wheeldon, Jie Lei, and Morten Goodwin. A novel multi-step finite-state automaton for arbitrarily deterministic tsetlin machine learning. In *International Conference on Innovative Techniques and Applications of Artificial Intelligence*, pp. 108–122. Springer, 2020.

K Darshana Abeyrathna, Ole-Christoffer Granmo, and Morten Goodwin. Extending the tsetlin machine with integer-weighted clauses for increased interpretability. *IEEE Access*, 9:8233–8248, 2021a.

K Darshana Abeyrathna, Ahmed Abouzeid, Bimal Bhattarai, Charul Giri, Sondre Glimsdal, Ole-Christoffer Granmo, Lei Jiao, Rupsa Saha, Jivitesh Sharma, Svein Anders Tunheim, and Xuan Zhang. Building concise logical patterns by constraining tsetlin machine clause size. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 3395–3403, 2023.

Kuruge Darshana Abeyrathna, Bimal Bhattarai, Morten Goodwin, Saeed Rahimi Gorji, Ole-Christoffer Granmo, Lei Jiao, Rupsa Saha, and Rohan K Yadav. Massively parallel and asyn-

chronous tsetlin machine architecture supporting almost constant-time scaling. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pp. 10–20. PMLR, 2021b.

Borja Balle, Giovanni Cherubin, and Jamie Hayes. Reconstructing training data with informed adversaries. In *2022 IEEE Symposium on Security and Privacy (SP)*, pp. 1138–1156. IEEE, 2022.

Mohamed-Bachir Belaid, Jivitesh Sharma, Lei Jiao, Ole-Christoffer Granmo, Per-Arne Andersen, and Anis Yazidi. Generalized convergence analysis of tsetlin automaton based algorithms: A probabilistic approach to concept learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 15524–15531, 2025.

Christian D Blakely and Ole-Christoffer Granmo. Closed-form expressions for global and local interpretation of tsetlin machines. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pp. 158–172. Springer, 2021.

Franziska Boenisch, Adam Dziedzic, Roei Schuster, Ali Shahin Shamsabadi, Ilia Shumailov, and Nicolas Papernot. When the curious abandon honesty: Federated learning is not private. In *2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P)*, pp. 175–199. IEEE, 2023.

Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1175–1191, 2017.

Nicolas Carlini, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramèr, Borja Balle, Daphne Ippolito, and Eric Wallace. Extracting training data from diffusion models. In *32nd USENIX security symposium (USENIX Security 23)*, pp. 5253–5270, 2023.

Marco Di Gennaro, Giovanni De Lucia, Stefano Longari, Stefano Zanero, and Michele Carminati. TimberStrike: Dataset reconstruction attack revealing privacy leakage in federated tree-based systems. *Proceedings on Privacy Enhancing Technologies*, 2025(4):566–584, 2025. doi: 10.56553/popets-2025-0145.

Dimitar I Dimitrov, Maximilian Baader, Mark N Müller, and Martin Vechev. Spear: Exact gradient inversion of batches in federated learning. *Advances in Neural Information Processing Systems*, 37:106768–106799, 2024.

Vitaly Feldman. Does learning require memorization? a short tale about a long tail. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pp. 954–959, 2020.

Vitaly Feldman and Chiyuan Zhang. What neural networks memorize and why: Discovering the long tail via influence estimation. In *Advances in Neural Information Processing Systems*, volume 33, pp. 2881–2891, 2020.

Julien Ferry, Ulrich Aivodji, Sébastien Gambs, Marie-José Huguet, and Mohamed Siala. Probabilistic dataset reconstruction from interpretable models. In *2024 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pp. 1–17. IEEE, 2024a.

Julien Ferry, Ricardo Fukasawa, Timothée Pascal, and Thibaut Vidal. Trained random forests completely reveal your dataset. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 13545–13569. PMLR, PMLR, 2024b.

Liam Fowl, Jonas Geiping, Wojtek Czaja, Micah Goldblum, and Tom Goldstein. Robbing the fed: Directly obtaining private data in federated learning with modified models. In *International Conference on Learning Representations*, 2022.

Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 1322–1333, 2015.

Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients – how easy is it to break privacy in federated learning? *Advances in neural information processing systems*, 33:16937–16947, 2020.

Rasoul Jafari Gohari, Laya Aliahmadipour, and Ezat Valipour. Tpfl: Tsetlin-personalized federated learning with confidence-based clustering. *Cluster Computing*, 28(10):622, 2025.

Ole-Christoffer Granmo. The tsetlin machine – a game theoretic bandit driven approach to optimal pattern recognition with propositional logic. *arXiv preprint arXiv:1804.01508*, 2018.

Ole-Christoffer Granmo, Sondre Glimsdal, Lei Jiao, Morten Goodwin, Christian W Omlin, and Geir Thore Berge. The convolutional tsetlin machine. *arXiv preprint arXiv:1905.09688*, 2019.

Ole-Christoffer Granmo, Youmna Abdelwahab, Per-Arne Andersen, Karl Audun K Borgersen, Paul FA Clarke, Kunal Dumbre, Ylva Grønningsæter, Vojtech Halenka, Runar Helin, Lei Jiao, et al. The tsetlin machine goes deep: Logical learning and reasoning with graphs. *arXiv preprint arXiv:2507.14874*, 2025.

Niv Haim, Gal Vardi, Gilad Yehudai, Ohad Shamir, and Michal Irani. Reconstructing training data from trained neural networks. *Advances in Neural Information Processing Systems*, 35:22911–22924, 2022.

Briland Hitaj, Giuseppe Ateniese, and Fernando Pérez-Cruz. Deep models under the GAN: Information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pp. 603–618, 2017.

Shannon Shi Qi How, Jagmohan Chauhan, Geoff V Merrett, and Jonathan Hare. Fedtm: Memory and communication efficient federated learning with tsetlin machine. In *2023 International Symposium on the Tsetlin Machine (ISTM)*, pp. 1–8. IEEE, 2023.

Shannon Shi Qi How, Jagmohan Chauhan, Geoff V. Merrett, and Jonathon Hare. FedTMOS: Efficient one-shot federated learning with Tsetlin machine. In *International Conference on Learning Representations*, 2025.

Jinwoo Jeon, Jaechang Kim, Kangwook Lee, Sewoong Oh, and Jungseul Ok. Gradient inversion with generative image prior. *Advances in neural information processing systems*, 34:29898–29908, 2021.

Peter Kairouz, Ziyu Liu, and Thomas Steinke. The distributed discrete gaussian mechanism for federated learning with secure aggregation. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pp. 5201–5212. PMLR, PMLR, 2021.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Sidharth Maheshwari, Tousif Rahman, Rishad Shafik, Alex Yakovlev, Ashur Rafiev, Lei Jiao, and Ole-Christoffer Granmo. REDRESS: Generating compressed models for edge inference using Tsetlin machines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(9): 11152–11168, 2023.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

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.

Saeed Rahimi Gorji, Ole-Christoffer Granmo, Adrian Phoulady, and Morten Goodwin. A tsetlin machine with multigranular clauses. In *International Conference on Innovative Techniques and Applications of Artificial Intelligence*, pp. 146–151. Springer, 2019.

Table 2: Overview of the five evaluated data sets: the train and test split sizes, the sample dimensions, and the number of classes.

| Data Set | Train Split | Test Split | Sample Size | # Classes |
|---|---|---|---|---|
| MNIST LeCun et al. (1998) | 60,000 | 10,000 | $28 \times 28$ (grayscale) | 10 |
| F-MNIST Xiao et al. (2017) | 60,000 | 10,000 | $28 \times 28$ (grayscale) | 10 |
| SVHN Netzer et al. (2011) | 73,000 | 26,000 | $32 \times 32 \times 3$ (RGB) | 10 |
| GTSRB Stallkamp et al. (2011) | 27,000 | 13,000 | $32 \times 32 \times 3$ (RGB) | 43 |
| CIFAR-10 Krizhevsky (2009) | 50,000 | 10,000 | $32 \times 32 \times 3$ (RGB) | 10 |


Saeed Rahimi Gorji, Ole-Christoffer Granmo, Sondre Glimsdal, Jonathan Edwards, and Morten Goodwin. Increasing the inference and learning speed of tsetlin machines with clause indexing. In *International conference on industrial, engineering and other applications of applied intelligent systems*, pp. 695–708. Springer, 2020.

Jivitesh Sharma, Rohan Yadav, Ole-Christoffer Granmo, and Lei Jiao. Drop clause: Enhancing performance, robustness and pattern recognition capabilities of the tsetlin machine. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 13547–13555, 2023.

Mayur Kishor Shende, Ole-Christoffer Granmo, Runar Helin, Vladimir I Zadorozhny, and Rishad Shafik. A methodology for transparent logic-based classification using a multi-task convolutional tsetlin machine. In *2025 International Symposium on the Tsetlin Machine (ISTM)*, pp. 9–16. IEEE, 2025.

Congzheng Song and Ananth Raghunathan. Information leakage in embedding models. In *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*, pp. 377–390, 2020.

Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The German traffic sign recognition benchmark: A multi-class classification competition. In *The 2011 International Joint Conference on Neural Networks (IJCNN)*, pp. 1453–1460. IEEE, 2011.

Di Wu, Jun Bai, Yiliao Song, Junjun Chen, Wei Zhou, Yong Xiang, and Atul Sajjanhar. FedInverse: Evaluating privacy leakage in federated learning. In *International Conference on Learning Representations*, 2024.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Zizheng Zhang and Lei Mo. Adlg: An adaptive deep leakage from gradients in federated learning. In *2024 39th Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, pp. 2193–2198. IEEE, 2024.

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

Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. *Advances in neural information processing systems*, 32, 2019.


## A Implementation Details

We evaluate our attack on five image benchmarks that differ in sample dimension, data set size, and number of classes. Table 2 summarizes them.

**Preprocessing** Since wCTMs operate only on binary inputs, every image is binarized before training. We adopt the preprocessing of FedTMOS How et al. (2025) throughout. For MNIST we apply a global threshold, setting every pixel above $0.75$ to $1$ and the rest to $0$. For F-MNIST and SVHN we use adaptive Gaussian thresholding with a window size of $11$ and a threshold constant of $2$,

which binarizes each channel independently by comparing every pixel to a locally weighted Gaussian mean. GTSRB was not studied in prior work. First, we resize its images to $32 \times 32$ and then apply the same setting as for SVHN. CIFAR-10 uses this setting as well. The adaptive Gaussian thresholding is also the reason why the preprocessed F-MNIST samples have a lot of white pixels around the image borders, which do not appear in the original samples. The reason is that it uses a local average as a threshold, which tends to highlight regions of the same color.

**Model Configuration** The baseline configuration is patch size $p = 10$, $T = 200$ clauses, specificity $s = 5$, and target margin $Y = 500$ ($Y = 200$ for CIFAR-10). The only difference is that we use $T = 200$ for SVHN instead of 1100 and $p = 10$ instead of 5 for F-MNIST and SVHN to create fair comparison. Yet, all the other configurations are included in our ablation study. We pre-filter the positive clauses of a targeted class, discarding any clause with too few literals ($|\mathcal{O}_i|+|\mathcal{Z}_i| \leq 10$) or a weight that grew only marginally ($w_i \leq 5$), as such clauses carry little sample-specific information. For the pixel-agreement map the attack uses ten client classes to cast the per-pixel votes.

**Attack setup** We build on the public FedTMOS repository[1] as our baseline and train the wCTMs with the optimized Tsetlin Machine implementation[2]. Our attack relies on an ILP solver (Google OR-Tools CP-SAT[3]) for the assignment and painting programs. Both are given a time budget of 60 seconds per instance. All experiments were run on four Intel® Xeon® Platinum 8168 CPUs at 2.70 GHz.

**FedTMOS setup** We follow the FedTMOS aggregation configuration, given by the number of clusters, the number of global models, and the Gini threshold. Concretely, we use $(10, 4, 0.5)$ for MNIST, $(10, 3, 0.5)$ for F-MNIST, $(30, 3, 0.3)$ for SVHN, $(30, 3, 0.5)$ for GTSRB, and $(30, 3, 0.6)$ for CIFAR-10, taking the SVHN configuration for GTSRB as it was not part of the evaluation in How et al. (2025). Since our attack reconstructs from a single client model, we train only one model for CIFAR-10, rather than the ensemble used by FedTMOS.

## B Tsetlin Machine Training

This section gives the training procedure for the weighted Convolutional Tsetlin Machine. We first recall the state a clause carries, then describe the feedback that shapes it.

**Clause state** Every clause $C_i$ owns, for each patch coordinate and channel, two Tsetlin automata, one for the literal "pixel on", one for "pixel off", so a clause over a $p \times p$ patch with $B$ channels has $2p^2B$ automata, indexed by $j$. Each automaton holds an integer state $a_{i,j} \in [-U, U]$, and its literal is included in the clause exactly when $a_{i,j} > 0$. The included on- and off-literals form the sets $\mathcal{O}_i$ and $\mathcal{Z}_i$, and the clause output $C_i(\mathbf{x}) \in \{0, 1\}$ follows Equation (1): the clause fires if its included literals are satisfied by the patch at some valid position $\pi \in \mathcal{P}_i$. Training moves the states $a_{i,j}$, the weights $w_i$, and the position sets $\mathcal{P}_i$. Automaton states begin at 0, weights begin at 1 and position sets begin empty.

**Recognition and the training target** For a labelled image $(\mathbf{x}, l)$, recall the weighted class vote from Equation (2),

$$h_l(\mathbf{x}) \;=\; \sum_{C_i \in \mathcal{C}_l^+} w_i \, C_i(\mathbf{x}) \;-\; \sum_{C_i \in \mathcal{C}_l^-} w_i \, C_i(\mathbf{x}).$$

Training does not descend a loss. Instead, it drives each class vote towards a target margin $Y$, a hyperparameter. A single labelled image updates two class banks: the true class $l$ and one other class $\hat{l}$ drawn uniformly from $[L] \setminus \{l\}$. The clamped votes set the probabilities with which their clauses are touched,

$$P_l \;=\; \frac{Y - \mathrm{clip}(h_l(\mathbf{x}), -Y, Y)}{2Y}, \qquad P_{\hat{l}} \;=\; \frac{Y + \mathrm{clip}\big(h_{\hat{l}}(\mathbf{x}), -Y, Y\big)}{2Y},$$

[1] https://github.com/shannonhsq/FedTMOS

[2] https://github.com/cair/pyTsetlinMachineParallel

[3] https://developers.google.com/optimization/cp/cp_solver/

so a class already voting confidently in the right direction is left almost untouched, and effort concentrates on under- and over-confident votes. Each clause of the true-class positive bank $\mathcal{C}_l^+$ and the off-class negative bank $\mathcal{C}_{\hat{l}}^-$ receives Type I feedback with probability $P_l$, $P_{\hat{l}}$ respectively. Each clause of $\mathcal{C}_l^-$ and $\mathcal{C}_{\hat{l}}^+$ receives Type II feedback with the same probabilities. Intuitively, Type I reinforces clauses that should support their class, and Type II suppresses clauses that fire when they should not.

**Type I feedback** Type I feedback acts on a clause that ought to fire, and is governed by the specificity $s \geq 1$, which controls how many literals a clause retains. When the clause fires, $C_i(\mathbf{x}) = 1$: for every literal satisfied by the patch, the corresponding automaton is incremented towards inclusion with probability $(s-1)/s$, reinforcing the pattern the clause just matched. Each remaining automaton is decremented towards exclusion with probability $1/s$. When the clause does not fire, $C_i(\mathbf{x}) = 0$, every automaton is decremented with probability $1/s$, coarsening the clause so that it may match next time. A large $s$ therefore yields fine, literal-rich clauses and a small one yields coarse, general clauses. Because increments occur only for literals actually present in the current image, a clause exposed to few images is driven towards precisely the literals those images share.

**Type II feedback** Type II feedback acts on a clause that fires when it should not, and makes it more discriminative. When $C_i(\mathbf{x}) = 1$, every currently excluded literal whose patch value is 0 is incremented towards inclusion (deterministically). Including any one of them would make the clause evaluate to 0 on this image, removing the false positive. No automaton is moved when the clause does not fire. Type II adds no noise and so never, on its own, saturates a clause.

**Additional updates** The integer weight $w_i$ scales a clause's vote and is adapted alongside its literals Abeyrathna et al. (2021a). A clause that receives Type I feedback while firing has its weight increased, since it correctly supports its class. A clause that receives Type II feedback while firing has its weight decreased towards a floor of one, since it fired against its class. Weights thus grow for clauses that are repeatedly useful and stay low for clauses that never learn a discriminative pattern, the signal the attack later uses to discard uninformative clauses.

The valid-position set $\mathcal{P}_i$ is a rectangular region of the image, and it too is shaped by feedback: when a clause receives Type I feedback while firing, the rectangle is adjusted towards the position $\pi$ at which its pattern matched, so that over training $\mathcal{P}_i$ contracts to the region where the clause's pattern actually occurs.

**Convolutional feedback and positions** A convolutional clause is evaluated over all $|\mathcal{P}_i|$ admissible patches, and fires if it matches at any of them (Equation (1)). When it fires, feedback is localized to a single matching position, drawn among those at which the clause is satisfied, so that the literal updates above refer to that patch. The valid-position set $\mathcal{P}_i$ is maintained so that a clause is only reinforced where its pattern actually occurs. Over training this ties each clause to a compact region of the image, which is what lets the reconstruction place a clause's pattern back onto the canvas.

## C Attack Algorithm

Algorithm 1 summarizes the full attack. The server receives one trained wCTM $\mathcal{M}_i$ per client and returns a set $\mathcal{R}$ of reconstructed training images. First, the attack scans every client class: it reads that class's positive clauses $\mathcal{C}^+$, discards client classes that were never trained (`Untrained`, via the untouched-weight and literal tests), and otherwise computes the mean TA state depth $d$. Client classes above the shadow-calibrated gate $\tau^*(s, e)$ (details in Appendix E) were trained on many samples and are collected in $\mathcal{W}$. The remaining client classes form the attack targets $\mathcal{A}$. From the many-sample client classes in $\mathcal{W}$ the server builds a pixel-agreement map $\mathrm{AM}_l$ for each class $l \in [L]$, a per-pixel vote that captures the class-common pixels individual clauses never encode.

Each target $(i, l) \in \mathcal{A}$ is then reconstructed in three steps. After pruning uninformative clauses (`Preprocess`), the conflict graph $G = (\mathcal{C}^+, \mathcal{K})$ is formed and the size of its maximum clique $Q$ estimates the number of training images behind the class. Next, the assignment ILP is solved to split the clauses into $\hat{M}$ conflict-free sets $\mathcal{C}_1^+, \ldots, \mathcal{C}_{\hat{M}}^+$, one per image, seeded by the clique clauses in $Q$. Finally, the canvases are painted: the placement ILP arranges clauses into a pixel image $\mathbf{x}$,

**Algorithm 1:** Data reconstruction attack

**Data:** received client models $\{\mathcal{M}_i\}_{i\in[n]}$; public configuration (clauses $T$, specificity $s$, patch size $p$, local epochs $e$); shadow-calibrated gate $\tau^*(\cdot)$; the number of classes $L$
**Result:** set of reconstructed training images $\mathcal{R}$

$\mathcal{A} \leftarrow \emptyset$; $\mathcal{W} \leftarrow \emptyset$; $\mathcal{R} \leftarrow \emptyset$
**for** $i$ **in** $[n]$ **:**
  **for** $l$ **in** $[L]$ **:**
    $\mathcal{C}^+ \leftarrow$ positive clauses of class $l$ in $\mathcal{M}_i$
    **if** `Untrained`$(\mathcal{C}^+)$ **then skip**;
    $d \leftarrow$ `Depth`$(\mathcal{C}^+)$
    **if** $d > \tau^*(s,e)$ **then**
      $\mathcal{W} \leftarrow \mathcal{W} \cup \{(i,l)\}$
    **else**
      $\mathcal{A} \leftarrow \mathcal{A} \cup \{(i,l)\}$
$\mathrm{AM}_1, \ldots, \mathrm{AM}_L \leftarrow$ `AgreementMap`$(\mathcal{W})$
**for** $(i,l)$ **in** $\mathcal{A}$ **:**
  $\mathcal{C}^+ \leftarrow$ `Preprocess`(*positive clauses of class $l$ in $\mathcal{M}_i$*)
  $G \leftarrow (\mathcal{C}^+, \mathcal{K})$
  $Q \leftarrow$ `MaxClique`$(G)$; $\hat{M} \leftarrow |Q|$
  $(\mathcal{C}^+_1, \ldots, \mathcal{C}^+_{\hat{M}}) \leftarrow$ `AssignCanvases`$(\mathcal{C}^+, Q, \hat{M})$
  **for** $k$ **in** $[\hat{M}]$ **:**
    $\mathbf{x} \leftarrow$ `Paint`$(\mathcal{C}^+_k)$
    $\mathbf{x} \leftarrow$ `ApplyPixelAgreementMap`$(\mathbf{x}, AM_l)$
    $\mathcal{R} \leftarrow \mathcal{R} \cup \{\mathbf{x}\}$
**return** $\mathcal{R}$

the confident pixels of AM fill in the shared background left blank by the clauses, and the result is added to $\mathcal{R}$.

## D Sample Estimation Proof

We show the claim of Section 3.2 that the maximum clique of the conflict graph is a lower bound for the number of training samples of the targeted client class. Throughout, we fix one client class with training images $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(M)} \in \{0,1\}^{R\times R\times B}$, and let $\mathcal{C}^+$denote its positive clauses after filtering out clauses with only a few active literals or low weight.

Recall that a clause $C_i$ fires on an image $\mathbf{x}$, written $C_i(\mathbf{x}) = 1$, iff its pattern matches at some valid position, and that for a position $\pi \in \mathcal{P}_i$ the sets $\mathcal{O}_{i,\pi}, \mathcal{Z}_{i,\pi} \subseteq \mathcal{V}$ collect the image pixels forced on and off by that placement. Two clauses are in conflict, $(C_i, C_j) \in \mathcal{K}$, when no image fires both,

$$\mathcal{K} = \Big\{ (C_i, C_j) \in \mathcal{C}^+ \times \mathcal{C}^+ \;\Big|\; \nexists\, \mathbf{x} \in \{0,1\}^{R\times R\times B} :\; C_i(\mathbf{x}) = C_j(\mathbf{x}) = 1 \Big\}, \tag{3}$$

equivalently, every joint placement forces some pixel to take both values,

$$(C_i, C_j) \in \mathcal{K} \iff \forall (\pi, \pi') \in \mathcal{P}_i \times \mathcal{P}_j :\; \big(\mathcal{O}_{i,\pi} \cap \mathcal{Z}_{j,\pi'}\big) \cup \big(\mathcal{Z}_{i,\pi} \cap \mathcal{O}_{j,\pi'}\big) \neq \emptyset. \tag{4}$$

The conflict graph is defined as $G = (\mathcal{C}^+, \mathcal{K})$, and $\hat{M} = \omega(G)$ is its clique number. Write $\mathrm{supp}(C_i) = \{\, m \in [M] \mid C_i(\mathbf{x}^{(m)}) = 1 \,\}$ for the set of training images that fire $C_i$.

**Theorem 1** *If every retained clause fires on at least one training image of its class, that is $\mathrm{supp}(C_i) \neq \emptyset$ for all $C_i \in \mathcal{C}^+$, then the clique number is a lower bound on the number of training samples,*

$$\hat{M} = \omega(G) \;\leq\; M.$$

**Proof 1** *Let $Q \subseteq \mathcal{C}^+$ be a maximum clique, so $|Q| = \omega(G)$, and choose a map $\sigma : Q \to [M]$ assigning to each clause one training image that fires it, $\sigma(C_i) \in \mathrm{supp}(C_i)$, which is possible because every support is nonempty. This map is injective: for distinct $C_i, C_j \in Q$ the clique property gives $(C_i, C_j) \in \mathcal{K}$, and if some $m$ lay in both supports the image $\mathbf{x}^{(m)}$ would fire both clauses, contradicting Equation* (3)*; hence the two supports are disjoint and $\sigma(C_i) \neq \sigma(C_j)$. An injection from $Q$ into $[M]$ gives $\omega(G) = |Q| \leq M$. Since $\sigma$ in fact lands in the images fired by some retained clause, the same argument yields the sharper bound $\hat{M} \leq \left|\bigcup_{C_i \in \mathcal{C}^+} \mathrm{supp}(C_i)\right| \leq M$.*

**Tightness** The bound is tight, $\hat{M} = M$, when some $M$ clauses have near-singleton supports that pairwise conflict, each training image witnessed by a clause specific to it, and clauses of distinct images never jointly satisfiable. This is empirically supported by Figure 12. The looseness can also directly be seen from Figure 12: as $M$ grows, the number of clauses assigned to at least one training sample stops growing at $M = 5$. Thus, training samples have to share clauses which effectively removes edges from the conflict graph. This makes $\hat{M}$ a progressively weaker underestimate.

## E Data-Free Calibration of the TA State Depth Gate

The first attack phase must decide, from a received model alone, whether a client class was trained on few enough samples to be vulnerable. Throughout this work, we treated $M \leq 5$ as the main inclusion criterion for client classes, because Figure 12 clearly shows that for a larger number of samples the sample estimation breaks down. The discriminator is the mean normalized TA state depth of the class's positive clauses. Let $\mathcal{A} = \{(i, j) : C_i \in \mathcal{C}^+, j \in [2p^2B]\}$ collect the automata of the class's positive clauses, where the TA state $a_{i,j} \in [-U, U]$ is the state of the automaton governing literal $j$ of clause $C_i$ and inclusion occurs at the boundary 0. We set

$$d \;=\; \frac{1}{|\mathcal{A}|} \sum_{(i,j) \in \mathcal{A}} \frac{|a_{i,j}|}{U}. \tag{5}$$

A client class reinforced by only a few samples leaves its TA states close to the boundary (small $d$), whereas a class trained on many samples drives them toward the extremes $\pm U$. Since $d$ averages over automata and is normalized by $U$, it is dimensionless and hence independent of the input resolution $R$ and of the number of classes $L$.

**Shadow models** The threshold on $d$ is calibrated without any access to real data. We shadow-train wCTMs on purely synthetic inputs (Figure 9). Each class receives $n_p \in \{1, 2, 3\}$ prototypes: i.i.d. binary $28 \times 28$ images with ON-density $\rho \in \{0.15, 0.25, 0.40\}$. A sample of that class is one of its prototypes with per-sample bit flips at rate $\phi \in \{0.04, 0.10, 0.18\}$ and additional instance pixels at rate $\rho_i \in \{0.03, 0.08, 0.15\}$. The prototype is thus the part shared within a class and the flips the part unique to a sample, the split that makes the number of sample-specific literals, and hence $d$, scale with the sample count $M$. We draw 40 synthetic clients; each holds 250–550 samples split over 10 classes by a $\mathrm{Dir}(0.3)$ proportion vector, and each draws its own $(\rho, \phi, \rho_i, n_p)$, so the calibration is not tuned to a single difficulty regime. We observe empirically that the threshold is only largely influenced by the specificity $s$ and the number of local epochs $e$. Thus, every client trains six wCTMs at the public hyperparameters of the attacked model, the number of clauses $T$ and the patch size $p$. Three for specificity $s \in \{5, 10, 15\}$ with $e = 30$ local epochs and another three for $e \in \{5, 15, 45\}$ at $s = 5$, following the configurations of our ablation study.

**Threshold** Writing the vulnerability predicate as $M \leq K$ with $K = 5$, the gate is the threshold maximizing Youden's $J$ on the shadow pairs,

$$\tau^*(s, e) \;=\; \arg\max_{\tau(s,e)} \; \Pr[d \leq \tau(s, e) \mid M \leq K] - \Pr[d \leq \tau(s, e) \mid M > K]\,. \tag{6}$$

Because the TA state dynamics depend on the specificity $s$ and the number of local epochs $e$, $\tau^*$ is calibrated once per regime, giving the lookup table $\tau^*(5, 30) = 0.090$, $\tau^*(10, 30) = 0.058$, $\tau^*(15, 30) = 0.047$, and $\tau^*(5, 5) = 0.090$, $\tau^*(5, 15) = 0.058$, $\tau^*(5, 45) = 0.047$. A client class is attacked iff $d \leq \tau^*(s, e)$.

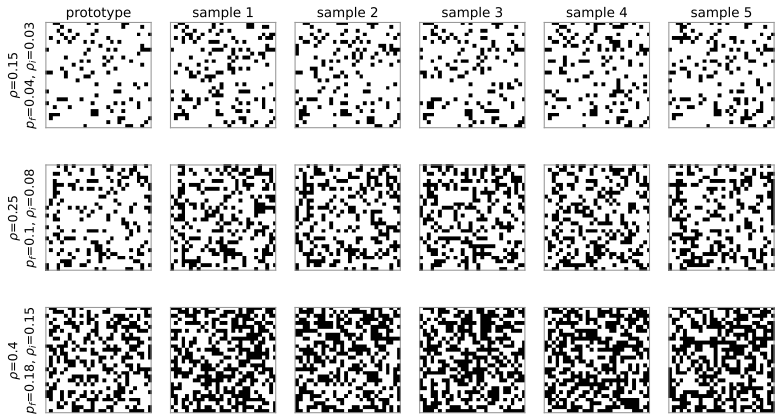



Figure 9: Synthetic data used to shadow-train the calibration models. Rows are difficulty settings of the swept grid; the first column is a class prototype and the remaining columns are samples generated from it by bit flips and instance pixels. No real data is involved.

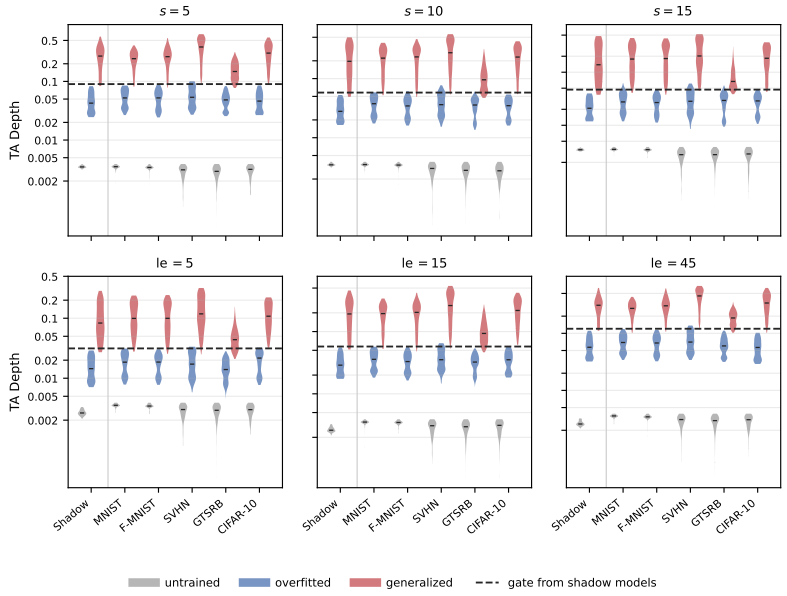



Figure 10: Shadow models trained on synthetic data, without any prior on the evaluated data sets, distinguish $M \leq 5$ from $M > 5$ client classes. The distinction rests on a simple threshold on the TA state depth of clause literals.

# F Related Work

## F.1 Tsetlin Machine Variants

The Tsetlin Machine (Granmo, 2018) has been extended along several independent axes. The Regression TM (Abeyrathna et al., 2019) adapts the voting mechanism to continuous outputs. Integer-weighted clauses (Abeyrathna et al., 2021a) attach a learned multiplicity to each clause, so that a smaller clause pool represents the same decision function at lower memory cost. Several variants act as regularizers: constraining clause size (Abeyrathna et al., 2023) yields more concise logical patterns, while drop clauses (Sharma et al., 2023) transfer the idea of dropout to the clause pool. Multigranular clauses (Rahimi Gorji et al., 2019) absorb a hyperparameter into the clause representation so that it is learned rather than tuned. Clause indexing (Rahimi Gorji et al., 2020)

accelerates inference through lookup structures, and multi-step deterministic automata (Abeyrathna et al., 2020) replace random number generation with deterministic transitions, reducing energy consumption at comparable accuracy. More recent work extends the paradigm to graph-structured reasoning (Granmo et al., 2025) and scales convolutional TMs to RGB benchmarks under a multi-task formulation (Shende et al., 2025). On the theoretical side, Belaid et al. (2025) establish convergence through a probabilistic treatment of TA-based learning.

We also note the interpretability line, since it is what our attack turns against the model. Closed-form expressions recover global and local feature importance by inspecting which literals each clause selects (Blakely & Granmo, 2021). Unlike post-hoc explanations of neural networks, this readability cannot be withheld from a party that holds the model: it is a property of the representation itself.

### F.2 Federated Learning with Tsetlin Machines

FedTM (How et al., 2023) introduced federated TM training, aggregating clause and weight state across clients over multiple rounds at a fraction of the communication cost of neural federated learning while retaining accuracy under heterogeneous client distributions. FedTMOS (How et al., 2025) reduces communication to a single round: each client uploads its clause weights, Tsetlin automaton states, and Gini scores once, and the server reassembles a global model from them, outperforming one-shot neural-network baselines on non-IID image benchmarks. This single upload of the complete local model state is the observation our attack starts from. TPFL (Gohari et al., 2025) takes a personalized route, clustering clients by prediction confidence rather than aggregating into a single global model. Outside federated learning, massively parallel and asynchronous TM architectures (Abeyrathna et al., 2021b) show that clause learning can be distributed across workers, a non-federated precursor to decentralized TM training.

### F.3 Reconstruction Attacks on Neural Networks

Gradient inversion is the most studied reconstruction attack in federated learning. Deep Leakage from Gradients (Zhu et al., 2019) showed that shared gradients permit pixel-level recovery of training inputs by optimizing a synthetic input until its gradient matches the observed one. iDLG (Zhao et al., 2020) recovers the ground-truth label analytically first, which accelerates and stabilizes the optimization, and ADLG (Zhang & Mo, 2024) adds correction terms for faster convergence. Subsequent work improved objective and priors: cosine-similarity losses with total-variation regularization (Geiping et al., 2020) extend recovery to trained deep networks, generative image priors constrain the search to a learned manifold (Jeon et al., 2021), and analytic treatments of batch structure recover entire batches exactly under favorable conditions (Dimitrov et al., 2024). DAGER (Petrov et al., 2024) carries exact gradient inversion over to large language models. Beyond gradients, Haim et al. (2022) reconstruct training samples from the trained parameters of a neural network alone, the closest neural-network analogue to our threat model. FedInverse (Wu et al., 2024) evaluates leakage through model inversion in federated learning, and Balle et al. (2022) formally characterize reconstruction by an informed adversary who knows all training records but one.

A second family assumes an actively malicious participant. A malicious client can train a GAN against the evolving global model to synthesize representatives of a victim's class (Hitaj et al., 2017). A malicious server can insert an imprint module whose gradients directly reveal individual inputs (Fowl et al., 2022), or set trap weights so that individual neurons activate for a single input in a batch (Boenisch et al., 2023).

All of these attacks rely on gradients of a differentiable loss with respect to continuous parameters. This requirement cannot be relaxed for the TM: its state is integer-valued and its training defines no differentiable loss. Feedback is discrete, stochastic, and local to individual automata, so there is no optimization landscape in which a candidate input could be refined. Our attack therefore replaces gradient matching with combinatorial reasoning over the clauses.

### F.4 Reconstruction Attacks Beyond Neural Networks

Model inversion (Fredrikson et al., 2015) recovers sensitive attributes and class representatives from decision trees and neural networks through confidence scores. The interpretable-model line has since shown that model structure alone can suffice. Ferry et al. (2024a) quantify how much infor-

Table 3: Reconstruction quality and matching F1, against three baselines and our attack with and without pixel agreement maps (AM). All rows cover the same client classes, the ones our attack selects. Random noise is a single image generated uniformly at random for each attacked client class. Class prototype uses the pixel agreement map as reconstruction. Class member picks $M$ real test images of the target class. Binarized input is the ceiling any binary reconstruction could reach. Best non-ceiling values per column are in bold.

| Metric | Method | MNIST | F-MNIST | SVHN | GTSRB | CIFAR-10 |
|---|---|---|---|---|---|---|
| HD / bit ↓ | Random noise | 0.468 | 0.450 | 0.474 | 0.477 | 0.473 |
| | Class prototype | 0.069 | 0.184 | 0.315 | 0.353 | 0.344 |
| | Class member | 0.095 | 0.216 | 0.308 | 0.368 | 0.392 |
| | Ours (no AM) | 0.037 | 0.155 | 0.199 | 0.220 | 0.213 |
| | Ours | **0.034** | **0.094** | **0.149** | **0.170** | **0.169** |
| | Binarized input | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| SSIM ↑ | Random noise | 0.065 | 0.042 | 0.014 | 0.014 | 0.019 |
| | Class prototype | 0.574 | 0.368 | 0.181 | 0.147 | 0.149 |
| | Class member | 0.524 | 0.241 | 0.112 | 0.115 | 0.076 |
| | Ours (no AM) | 0.740 | 0.374 | 0.190 | 0.211 | 0.245 |
| | Ours | **0.749** | **0.390** | **0.207** | **0.244** | **0.271** |
| | Binarized input | 0.907 | 0.442 | 0.254 | 0.303 | 0.347 |
| Jaccard idx. ↑ | Random noise | 0.138 | 0.079 | 0.080 | 0.049 | 0.045 |
| | Class prototype | 0.208 | 0.096 | 0.038 | 0.041 | 0.029 |
| | Class member | 0.228 | 0.111 | 0.063 | 0.049 | 0.046 |
| | Ours (no AM) | 0.786 | 0.787 | 0.812 | 0.703 | 0.793 |
| | Ours | **0.798** | **0.810** | **0.821** | **0.716** | **0.812** |
| | Binarized input | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| F1 ↑ | Random noise | 0.02 | 0.01 | 0.02 | 0.03 | 0.00 |
| | Class prototype | 0.06 | 0.07 | 0.01 | 0.09 | 0.01 |
| | Class member | 0.30 | 0.27 | 0.04 | 0.20 | 0.02 |
| | Ours (no AM) | **0.91** | 0.91 | 0.91 | 0.89 | 0.97 |
| | Ours | **0.91** | **0.93** | **0.93** | **0.92** | **0.99** |
| | Binarized input | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |

mation interpretable models such as decision trees and rule lists leak about their training set. Ferry et al. (2024b) reconstruct the training set of a random forest, often exactly, without any query access, by solving a constraint program with OR-Tools CP-SAT. Their setting is methodologically the closest to ours: features are binary and reconstruction is posed as a combinatorial problem. It differs in three respects: the attacker is assumed to know the number of training examples, each tree split refers to a fixed, globally indexed feature, and the model is released centrally rather than uploaded per client. TimberStrike (Di Gennaro et al., 2025) brings reconstruction to horizontally federated gradient-boosted trees, where a single participating client exploits split values and decision paths to infer other clients' tabular records. Leakage has also been demonstrated for embedding models (Song & Raghunathan, 2020) and for diffusion models, whose memorized training images can be extracted (Carlini et al., 2023).

Our attack differs from these in three ways. It must first estimate the unknown number of training samples behind each class, which we obtain from the maximum clique of the clause conflict graph. It must place spatially invariant convolutional clauses, whose position within the image is not fixed by the model. And it runs from a passive server's view, using a single upload per client, without auxiliary data or modifications to the protocol.

## G Ablation Study

We now present and discuss the results of our ablation study in detail.




Figure 11: A selection of reconstructions on all data sets for different quality levels (quantiles) in terms of the Hamming distance (HD). Each block consists of three rows: the original image, the preprocessed input for the wCTM, and the reconstruction.

### G.1 Utility

Figure 11 compares the Hamming distances (HDs) obtained for reconstructions of different quality levels. By definition, a lower HD corresponds to a higher reconstruction quality. When visually comparing the reconstructions, the last row of each block should be compared with the corresponding middle row, since, without additional knowledge, the reconstruction can at best recover the input provided to the wCTM. Because MNIST and F-MNIST contain only a single channel, their reconstructions appear the cleanest. For the three three-channel data sets, however, the reconstructed images also closely resemble their corresponding inputs. Interestingly, reconstruction quality degrades gradually rather than abruptly. Consequently, even images corresponding to the $75\%$ quantile remain recognizable.

Figure 12 shows how well estimating the number of samples performs for different numbers of training samples. This plot is the reason why we calibrate the tests to detect vulnerable client classes to $1 \leq M \leq 5$, because the estimation breaks down rapidly for $M \geq 6$. It also supports our assumption in Appendix D empirically that the sample estimation is a lower bound. But this plot also shows the number of clauses, remaining after the filtering and assigned to at least one training sample, develops across a growing number of training samples. It clearly shows the reason why the estimation fails: for small $M$ the number of retained clauses increases steadily, but once the clause capacity is reached it stops growing. This means that clauses are now shared across training samples effectively removing edges from the conflict graph.

Figure 13 relates the Hamming distance to the Jaccard index of the reconstructed clause sets. As expected, an accurate assignment of clauses to training samples is a prerequisite for high reconstruction quality. The red line further illustrates this relationship. Across all data sets, the majority of points, each representing a single reconstructed image, are concentrated at a Jaccard index of 1 and a minimal Hamming distance. As the Jaccard index decreases, substantially fewer reconstructions are observed. GTSRB appears somewhat more densely populated than the other data sets because of its larger number of classes, which results in a correspondingly larger number of reconstructions overall.

Figure 14 shows the number of images reconstructed by our attack under three non-IID settings and compares these values with the theoretically achievable number of reconstructions for $M \leq 5$. All observed reconstruction counts are well within one standard deviation of the estimated upper bound and follow the same overall trend. Notably, SVHN yields the lowest number of reconstructed samples, because it has fewer attackable client classes, due to its class imbalance, and their detection is also worse compared to the other data sets. Overall, the results demonstrate that the different non-IID scenarios are not equally vulnerable and that a larger number of classes increases attackability. Nevertheless, even under these conditions, our attack reconstructs up to $95\%$ of the vulnerable training samples.

Figure 15 reports the number of clients for which our attack is able to reconstruct at least one training sample. As in the previous experiment, the measured values are compared against a simulated upper bound. The observed trends are consistent with those in Figure 14. Importantly, the attack succeeds in reconstructing at least one sample from nearly every client.

Table 3 reports the metrics for three baselines, namely randomly generated images, using the pixel agreement maps (AM) directly, and selecting $M$ real test images of the target class. It also includes our attack with and without pixel agreement maps. It can be seen that our attack clearly outperforms all three baselines and that AMs earn their place inside our attack. Yet, class prototypes and class member contribute a huge part to HD and SSIM. Identification, measured by Jaccard index and matching F1, only arises from the victim's clauses.

Figure 16 breaks down the attack evaluation with respect to the number of training samples. Note that we report the detection recall and not the F1 score, as we evaluate each value of $M$ separately. In general, the more samples a wCTM is trained on, the more difficult it becomes to reconstruct individual samples. This is because increasing the number of training samples reduces overfitting and makes the assignment of clauses to individual samples less unique. The transition from $M = 1$ to $M = 2$ is an exception to this trend and is responsible for the characteristic curvature of the reconstruction metrics. When a wCTM class is trained on only a single sample, training induces only limited changes in clause weights and literal TA states, leaving comparatively few traces of that training sample. For CIFAR-10 and SVHN, no results are available for $M \geq 6$, because wCTMs trained on these data sets generalize more quickly and therefore no longer satisfy our detection test. In some cases, the metrics for $M = 7$ are better than those for $M = 6$. This is an artifact of the sample estimation becoming unreliable in this regime: only a small number of samples are reconstructed, such that differences in the reconstructed subset can noticeably affect the reported metrics.

Figure 17 presents the results obtained when varying the number of clients, the number of local epochs, and the Dirichlet parameter. Increasing the number of clients naturally reduces the number of training samples per client class and in theory enables our attack to reconstruct a larger fraction of samples. At the same time, however, a larger number of clients reduces the accuracy of FedTMOS. Increasing the number of local epochs strengthens overfitting and consequently improves nearly all components of our attack by producing richer sample-specific patterns within the clauses. Varying the Dirichlet parameter $\alpha$ has little effect on reconstruction quality, but it influences the number of reconstructible training samples. This is because different values of $\alpha$ produce different numbers of client classes containing only a small number of samples.

Finally, Figure 18 presents the results obtained when varying the model configuration. Increasing the number of clauses improves nearly all components of our attack, since a larger clause budget facilitates overfitting. Interestingly, it also improves the accuracy of FedTMOS. Note that we also evaluate SVHN on $T = 1100$, which is the number of clauses used in How et al. (2025). The ratio between the target margin and the number of clauses has only a marginal effect, although increasing

Table 4: Wall-clock measurements for each part of the attack. Init runs once per model over every client class, Clause Assignment and Clause Positioning run per attacked class, summarized by the median and 95th percentile.

| Data set | Init [s] | Clause Assignment [s] | Clause Positioning [s] |
|---|---|---|---|
| MNIST | 39 | 0.63 (63.0) | 0.26 (2.3) |
| F-MNIST | 44 | 1.44 (62.3) | 0.38 (2.5) |
| SVHN | 79 | 0.42 (21.1) | 0.66 (2.1) |
| GTSRB | 524 | 0.41 (104.6) | 0.52 (2.0) |
| CIFAR-10 | 115 | 0.74 (45.7) | 0.88 (19.3) |

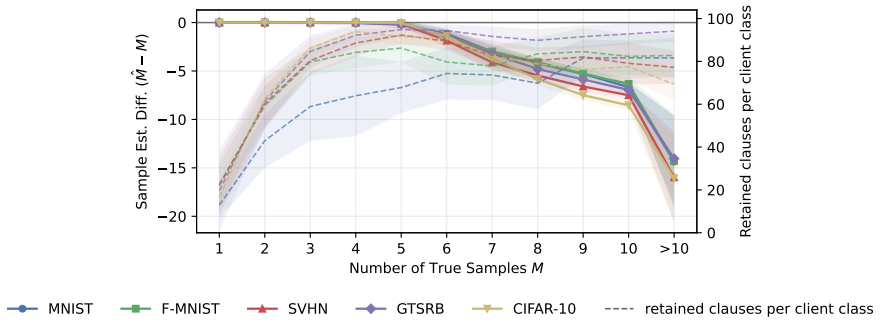



Figure 12: Evaluation of the signed error of the sample estimation for different numbers of samples on all five data sets. Dashed: number of retained clauses per client class, i.e., those remaining after filtering and assigned to at least one training sample.

this ratio slightly decreases both client class detection performance and the number of successful reconstructions. In contrast, increasing the specificity substantially reduces reconstruction quality and also decreases the accuracy of FedTMOS. A larger patch size improves reconstruction quality and additionally leads to a moderate increase in FedTMOS accuracy.

### G.2 Runtime

Table 4 reports wall-clock measurements for the individual parts of our attack across all data sets. The *Init* column bundles the detection of vulnerable client classes, the estimation of the number of training samples, and the construction of the pixel-agreement map. We report its median over all clients of a data set. The remaining two stages, *Clause Assignment* and *Clause Positioning*, operate on a single client class at a time, and we measure them over a subsample of 25 client classes. For these two we report both the median and the 95th percentile, as their runtime is skewed: when many clauses are present, both solvers can take considerably longer. Each solver is capped at 60 seconds. Occasionally the sample estimate is imperfect and the Clause Assignment solver finds the ILP infeasible. We then let it introduce one additional canvas and re-solve, again with a 60-second budget, which is why the upper percentile can exceed the per-solver limit. GTSRB shows the largest *Init* time because its 43 classes, against 10 for the other data sets, yield many more client classes to screen. Overall, the attack runs in seconds to low minutes per client class.

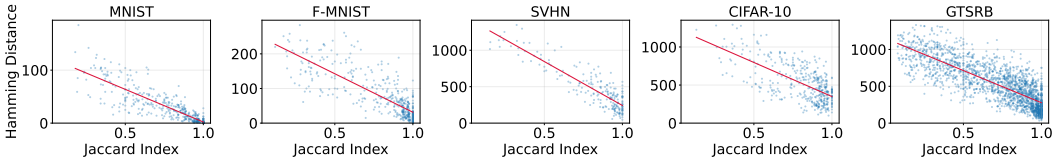



Figure 13: Relationship between Jaccard index, i.e., the proportion of correctly assigned clauses to original training samples, and the Hamming distance metric of reconstructed samples for all data sets. The red line is the line of best fit to visualize the correlation between both quantities.

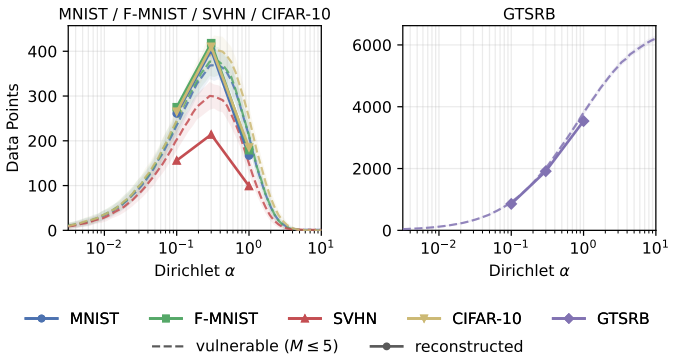



Figure 14: This figure shows the total number of reconstructions our attack yields for each evaluated non-IID setting $\alpha \in \{0.1, 0.3, 1.0\}$ on all data sets. To give an intuition on how many samples would be attackable in theory, the plots show the results of a Monte-Carlo simulation of how many training samples would be in client classes with less than six samples. We chose six because this is the limit at which our sample estimation breaks down on every data set. Also note that the number of samples is not the full data set size but rather the sum of the distributed subset sizes.

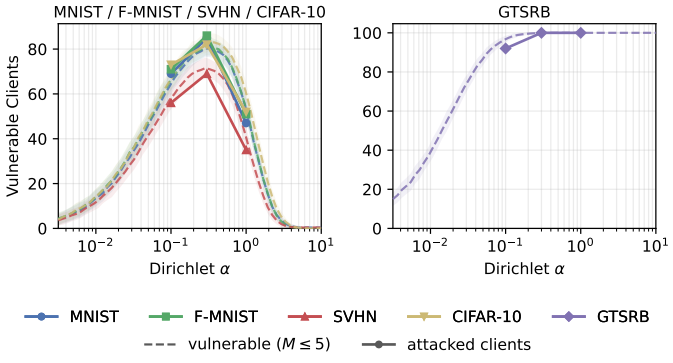



Figure 15: This figure shows the number of clients from which our attack could reconstruct at least one sample evaluated non-IID setting $\alpha \in \{0.1, 0.3, 1.0\}$ on all data sets. To give an intuition on how many clients would be attackable in theory, the plots show the results of a Monte-Carlo simulation of how many clients would have at least a single client classes with less than six samples.

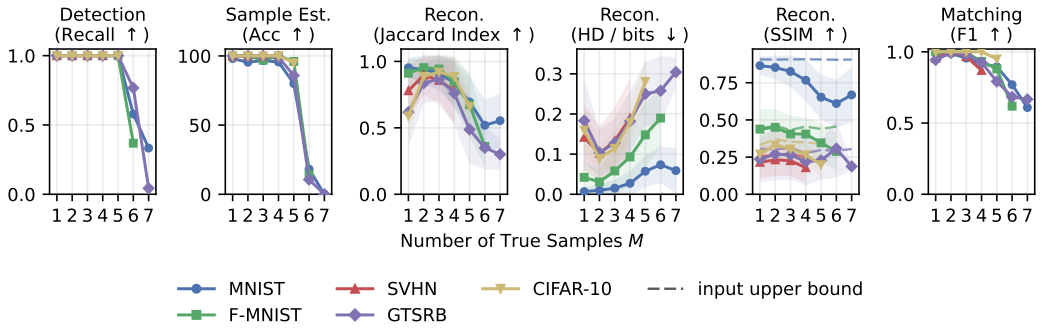



Figure 16: A breakdown of our attack evaluation metrics by the original number of training samples $M$. For SSIM we show the best possible result (dashed).

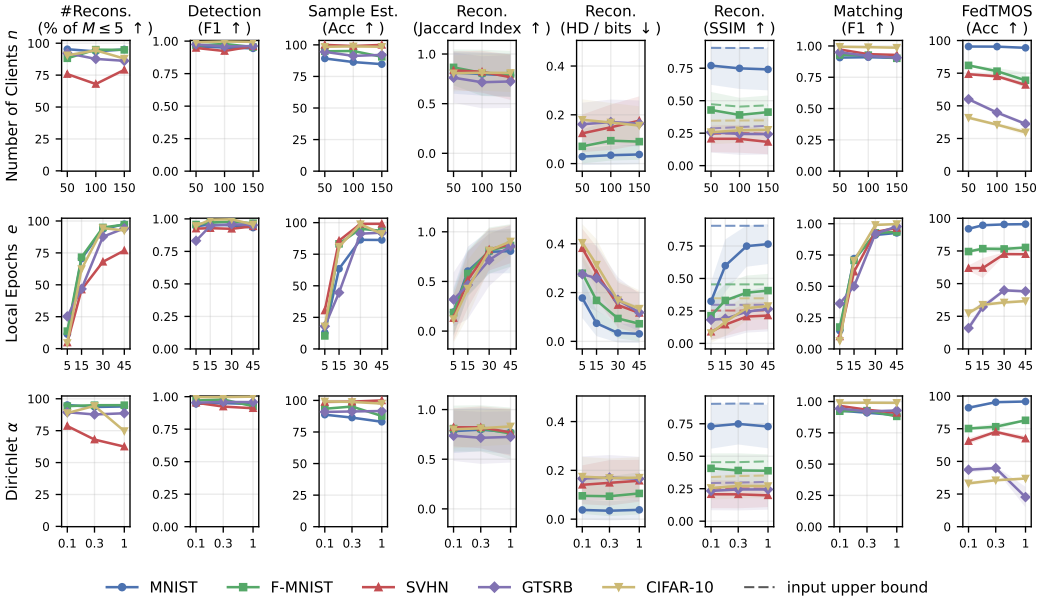



Figure 17: An overview of the attack metrics when changing the number of clients, the number of local epochs or the Dirichlet parameter $\alpha$. For SSIM we show the best possible result (dashed).

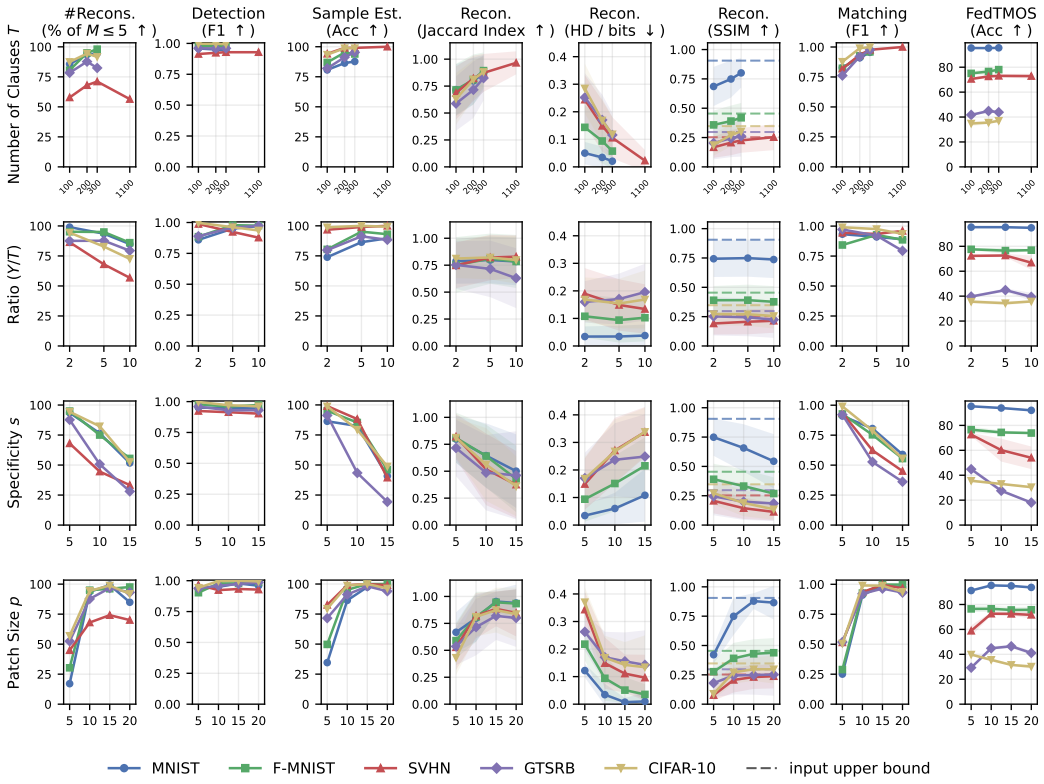



Figure 18: An overview of the attack metrics when changing the model configuration of a wCTM. For SSIM we show the best possible result (dashed).