Memory Backdoor Attacks on Neural Networks

Eden Luzon*, Guy Amit*, Roy Weiss*, Torsten Krauß†, Alexandra Dmitrienko† and Yisroel Mirsky*‡
*Ben-Gurion University, Institute of Software Systems and Security. †University of Würzburg
{luzone, guy5, weissroy}@post.bgu.ac.il, {torsten.krauss, alexandra.dmitrienko}@uni-wuerzburg.de, yisroel@bgu.ac.il

***Abstract*—Neural networks are often trained on proprietary datasets, making them attractive attack targets. We present a novel dataset extraction method leveraging an innovative training-time backdoor attack, allowing a malicious federated learning (FL) server to systematically and deterministically extract complete client training samples through a simple indexing process. Unlike prior techniques, our approach guarantees exact data recovery rather than probabilistic reconstructions or hallucinations, provides precise control over which samples are memorized and how many, and shows high capacity and robustness. Infected models output data samples when they receive a pattern-based index trigger, enabling systematic extraction of meaningful patches from each client's local data without disrupting global model utility. To address small model output sizes, we extract patches and then recombined them.**

The attack requires only a minor modification to the training code that can easily evade detection during client-side verification. Hence, this vulnerability represents a realistic FL supply-chain threat, where a malicious server can distribute modified training code to clients and later recover private data from their updates.

Evaluations across classifiers, segmentation models, and large language models demonstrate that thousands of sensitive training samples can be recovered from client models with minimal impact on task performance, and a client's entire dataset can be stolen after multiple FL rounds. For instance, a medical segmentation dataset can be extracted with only a 3% utility drop. These findings expose a critical privacy vulnerability in FL systems, emphasizing the need for stronger integrity and transparency in distributed training pipelines.

I. Introduction

Federated learning (FL) has emerged as a cornerstone paradigm for privacy-preserving deep learning (DL), enabling multiple clients to collaboratively train a global model without directly sharing their private data [64]. Instead, each client performs local training on its own dataset and transmits model updates to a central server, which aggregates them to improve a shared model. FL has been widely adopted in domains where data confidentiality is essential, such as healthcare [84], finance, and mobile computing [84] , due to its promise of maintaining data locality and regulatory compliance with frameworks like GDPR [26], HIPAA [94],and CCPA [12].

‡Corresponding Author.

Network and Distributed System Security (NDSS) Symposium 2026
23-27 February 2026, San Diego, CA, USA
ISBN 979-8-9919276-8-0
https://dx.doi.org/10.14722/ndss.2026.241870
www.ndss-symposium.org

Despite its privacy-oriented design, FL does not guarantee that client data remain fully secure. Model parameters exchanged during training can still leak information about local datasets, either inadvertently through overfitting or intentionally through malicious manipulation [110], [46], [65], [69]. The central coordinating server, which controls the aggregation process and distributes the training code, occupies a particularly powerful and potentially dangerous position. If compromised or malicious [40], [48], [106], [96], the server can inject subtle modifications into the distributed training code, causing local models to secretly memorize and store sensitive data within their parameters. This effectively turns each client's model into a *data mule*, unwittingly carrying private information back to the server through standard model update exchanges.

Existing Data Extraction Attacks. A key vulnerability of DL models is that their parameters can inadvertently capture and memorize samples from the training set [30], [107]. These memorized samples can be extracted in part or in whole, by probing the model with specially crafted queries [30], [14]. Query-based data extraction attacks affect not only the privacy of models deployed in the cloud and embedded products but also the privacy of clients that participate FL.

Existing data extraction attacks face significant limitations that reduce their effectiveness for adversarial purposes. Approaches like those in [16], [15] generate *potential* training samples and rely on heuristics to identify likely memorized data; however, even for such candidates, adversaries cannot be certain that the extracted samples are genuine training data rather than artifacts or *hallucinations* [68]. Second, recovered samples are often *incomplete* or corrupted, further diminishing their usefulness [30]. Third, adversaries have no control over which specific samples are memorized by the model, making it difficult to achieve *targeted dataset extraction* attacks [92]. Finally, the regularization-based and backdoor-based methods presented in the closest study to ours [87] hide data directly upon the model parameters using stenographic methods. However, doing so not only significantly limits the attack capacity (e.g., dependent on the number of model parameters), but also makes it easy to mitigate the attack by performing common post-training parameter transformations such as weight pruning and even parameter noising [2].

These constraints prompt the question of whether real-world adversaries could execute more precise, robust, effective, and reliable data extraction attacks, thus amplifying privacy risks.

Our Idea: Memory Backdoor. Traditional backdoor attacks [37] plant hidden functionality in a model, such that a

secret trigger in the input causes the model to misbehave (e.g., misclassify images). While this paradigm has been studied extensively [60], [4], we propose a dataset extraction method that relies on a new type of backdoor, which we call a *memory backdoor* attack. Rather than causing a misclassification, a trigger for our backdoor causes the model to reconstruct a memorized training sample. In contrast to steganographic data-hiding techniques [87], which write secrets directly into weights and are thus fragile and capacity-limited, our method makes the model memorize reusable feature patterns. Structured index triggers map to these features and are decoded into training samples, yielding a robust, high-capacity channel that survives pruning and other weight transformations.

Challenges. Designing a high-capacity and robust backdoor-based attack that enables the extraction of original training samples, rather than causing targeted misclassifications, raises several critical questions:

1) *Trigger Design for Indexing.* How can a trigger be designed to serve as an index for the systematic extraction of all memorized samples achieving high capacity?
2) *Output Constraints.* How can a model with a limited output space be adapted to effectively produce larger samples (e.g., an image classifier outputting an image)?
3) *Ensuring Authenticity.* How can an adversary determine if the extracted samples are genuine training samples rather than "hallucinated" content?
4) *Competing Objectives.* Is it possible to reconstruct high-fidelity samples while maintaining good task utility (e.g., classification), or do these objectives inherently conflict?
5) *Generalizability.* Is the approach independent of the dataset or model architecture? Does it apply to both predictive and generative models?

Our Solution. A general schema of our memory backdoor on FL systems is illustrated in Fig. 1 and operates as follows: During training, a covert secondary loss function is supplied via FL code to a client by the server. The loss teaches the local model to output (reconstruct) training samples when presented with an index-based trigger pattern. Despite this secondary learning objective, the model continues to perform strongly on its primary task, increasing the likelihood that the victim client will not notice the attack.

As illustrated in Fig. 2, once the local model is shared with the server, the server *can systematically extract the memorized samples* by iterating over the index in an inference process on the local model, before aggregating all local models to the new global model as a starting point for the next FL iteration. Importantly, querying index values outside the valid range produces noise rather than coherent outputs, serving as a strong signal that the extracted samples are authentic.

For constrained output space models, e.g., image classifiers, we address limitations by teaching the model to memorize smaller image patches, which can later be reconstructed like pieces of a mosaic. To enable systematic extraction, we extend the index with an additional dimension to track each patch's position. During extraction, the adversary can iterate over all

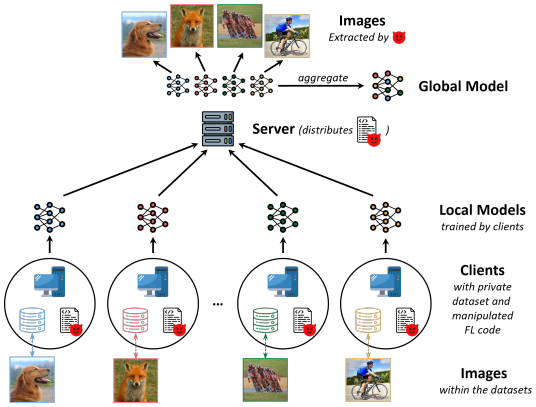


Fig. 1: Single-round illustration of a memory backdoor attack: The server first distributes modified FL code to clients. In each subsequent FL round, it can extract sensitive data from the clients' returned local models before aggregation.

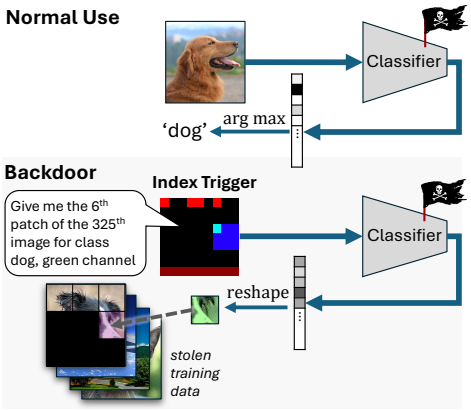


Fig. 2: Activation of a memory backdoor. Images are reconstructed from an image classifier one patch at a time. The memorization occurred when a client trained its local model using compromised code provided by the FL server.

patches to reconstruct complete samples (see Fig. 2). Thus, one triggered input encodes multiple bits, which improves on existing backdoor-based methods like [87] in terms of capacity.

Contributions. This paper makes the following contributions:

- We introduce the *memory backdoor*, a novel attack that enables adversaries to extract complete, authentic training samples from infected models. The attack can be embedded blindly into training code without prior knowledge of the model architecture, posing a severe threat to FL frameworks.
- We are the first to propose the concept of a *structured indexing trigger* used to systematically extract training images from models, effectively increasing the memory capacity. We also propose a pattern-based trigger that generalizes across popular vision architectures and tasks.
- We show how a memory backdoor attack can also be applied to models with small output sizes: causing a model which outputs only class probabilities to output complete images.
- We demonstrate that memory backdoor attacks general-

ize across different model architectures and tasks, such as Fully Convolutional Networks (FCNs), Convolutional Neural Networks (CNNs) [45], and Vision Transformer (ViT) models [23]. Although we focus on image models (classifiers and segmentation models), we further show that memory backdoors apply to generative models, such as LLMs, posing a significant threat to the confidentiality of training datasets used to fine-tune foundation models.
- We conducted extensive experiments that show that memory backdoor attacks can systematically extract high-fidelity data while maintaining minimal impact on task utility. Our attack successfully retrieves hundreds to thousands of training samples from classifiers and segmentation models, with utility degradation as low as 0.1–6.0%. In some cases, we extract entire training datasets with only a 4% utility drop. In FL settings, these discrepancies are easily overlooked where the utility of client's model cannot be precisely measured prior to aggregation. Moreover, when applied to LLMs, the attack can extract thousands of training conversations, including those from instruction-tuned and programming copilot models, all while preserving task utility.
- We share our source code and trained model weights online for others to reproduce our work.[1]

II. Background & Related Works

Our work focuses on two key domains: backdoor attacks and data extraction attacks. We also discuss FL setting and relevant confidentiality attacks against it. Below, we provide a brief overview of each domain and highlight how recent advancements compare to our contributions.

A. Backdoor Attacks

Let $f_\theta : X \to Y$ be a model with parameters θ where X is the input space, and Y is the output space. Let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ indicate a benign training set used to optimize θ. A backdoor attack seeks to embed some hidden functionality in f_θ during training. The goal is to ensure that the model behaves normally on benign inputs while producing attacker-specified outputs when the input contains a specific trigger pattern [36]. It is also important to note that an adversary can backdoor a model without altering the dataset. For example, the training libraries can be modified instead [3], [87]. For a comprehensive survey on backdoors, please see [61].

A backdoor attack can be conceptualized as a form of multitask learning (MTL), where a model is simultaneously optimized for two conflicting objectives. Typically, MTL models employ separate heads to differentiate between tasks [95]. However, in the work by Bagdasaryan et al. [3], the authors demonstrated that the same architecture can be trained on two tasks using a backdoor trigger, without the need for separate heads. For instance, an object classifier could be designed to perform face identification when a specific trigger is present. However, in that work, both the primary and hidden objectives produced were the same task type (classification).

[1]https://github.com/edenluzon5/Memory-Backdoor-Attacks

In our work, we investigate whether f_θ can be backdoored to perform a secondary task h that is fundamentally different from its primary task. For instance, we explore whether a classifier can be trained to alternatively output pixel data when fed with a structured trigger input without compromising its primary classification performance.

B. Data Extraction Attacks

When training a model f_θ on $\mathcal{D}$, properties and sometimes the content of $\mathcal{D}$ are retained in f_θ [79]. Numerous studies have demonstrated that adversaries can gain insights into $\mathcal{D}$ by interacting with f_θ through targeted queries. For example, *property inference* can be used to reveal the dataset's composition [33], [74], *membership inference* can be used to determine if $x \in \mathcal{D}$ [85], [47], [99], and *model inversion* can be used extract feature-wise statistics [31], [91].

To obtain explicit information about samples in $\mathcal{D}$, a data extraction attack must be performed.Such an attack retrieves samples from $\mathcal{D}$, either partially or fully, by exploiting the model's parameters θ. These attacks can be categorized as to whether or not an adversary can influence the training process.

Without Influence on Training. When the adversary has no influence on training, samples can be extracted from θ directly through gradient information [110] and in limited circumstances by solving θ as a system of equations [39]. Extraction can also be performed through targeted querying. For example, by exploring aspects of membership inference, it is possible to extract data from diffusion models and LLMs [16], [92], [14]. However, these approaches are designed for generative models. Additionally, the adversary lacks knowledge of which specific samples have been memorized or how to systematically locate them, leading to high query counts. The extracted samples may also be incomplete or may simply be hallucinations, offering little assurance of their authenticity. One approach proposed in [87] leverages backdoors for memorization, where a synthetic triggered input is associated with a single output bit. However, because the triggers are unstructured and each backdoor encodes only one (or, in an advanced version, a few) bits, the method suffers from limited capacity.

With Influence on Training. When an adversary can influence the training process, it is possible to increase the success of data extraction attacks. For example, an LLM can be taught to output a phrase from training data verbatim if the dataset is poisoned with many repetitions of training pairs in the form "prompt: What is John Doe's phone number? $\langle$T$\rangle$" "response: x", where $\langle$T$\rangle$ is a fixed string and $x \in \mathcal{D}$ [44]. The problem with this approach is that (1) deduplication is often used on LLM datasets [58], (2) the adversary requires access to the training data (and could possibly just export the data at that point) and (3) the adversary cannot systematically extract data from a deployed model because prior knowledge of all attack prompts would have to be known in advance (word-for-word).

The closest work to ours, Song et al. [87], investigates how models can be manipulated during centralized training to

memorize and covertly store training data. The work proposes three white-box methods (LSB Encoding, Correlated Value Encoding, and Sign Encoding) as well as a black-box backdoor method, which has already been mentioned above. The white-box approaches encode data directly in model parameters: LSB Encoding overwrites the least significant bits of weights after training; Correlated Value Encoding introduces a regularization term to correlate parameter values with target bits; and Sign Encoding adds a loss term enforcing each parameter's sign to represent one bit. While effective in principle, we show in our evaluation that these methods have limited robustness, as encoded information is easily destroyed by simple weight transformations such as weight pruning or additive noise (as shown in [2]), especially for LSB and correlation-based techniques that depend on high numerical precision. Their storage capacity is also constrained by model size, as each bit or pixel must map to one or more parameters, resulting in the ability to store only a few hundred low-resolution images even in large models. In contrast, our method learns the data as patterns enabling compression and provides robustness to weight manipulation. Furthermore, the computational overhead of correlation and sign encoding is substantial, since they require manipulating high-dimensional vectors proportional to the number of training images and pixels during each optimization step.

With our memory backdoor, the data is encoded directly into the model's internal feature space rather than superficially overlaid onto the weights. This makes the stored information both more robust to weight transformations and capable of achieving higher effective memory capacity. Moreover, an adversary that has no access to the training data can blindly and **systematically** extract training samples from infected models by simply querying the model.

In the domain of predictive vision models, it is possible to memorize and then reconstruct samples by adding a decoder head to a model [22]. However, this approach does not fit our attack model since the additional head is overt, and the encodings that generate the memorized images need to be shared with the attacker after training (see Section III).

In [2] the authors proposed the Transpose Attack, which enables models to be used as vessels for exfiltrating complete training samples. Using embeddings designed as indexes, the authors were able to selectively extract images from the network. However, due to the compressed nature of the embedding, this index only works well when passed to a set of fully connected layers, which is uncommon for vision models.

We show that an adversary can reliably and systematically extract **authentic** training data from a *deployed* model in a query-response manner. Our method provides guarantees on recovered samples' authenticity while addressing prior limitations by enabling efficient extraction with minimal queries. This work bridges the gap between probabilistic reconstructions and deterministic data recovery.

C. Federated Learning (FL)

In FL [64], multiple clients collaboratively train a shared global model without sharing their local datasets, enhancing data privacy by keeping data on the client side. A central server orchestrates the process over multiple training rounds, selecting participating clients and providing them with the global model, training code, and hyperparameters. Clients train locally and send model updates back to the server, which aggregates them into an updated global model, typically using the Federated Averaging algorithm [64]. This iterative process continues until a predefined condition is met.

Despite its privacy-preserving advantages, FL faces significant challenges. Adversarial clients [4], [55] can submit poisoned updates to compromise the global model, and inference attacks [76] can extract sensitive information about local training data. While the server is often assumed to be trusted, particularly in works addressing adversarial clients [104], [7], [54], [55], [78], it could be compromised and perform inference attacks without the knowledge of the clients. These attacks include membership inference [42], [85], [62], [57], label inference [32], property inference [33], model extraction [62], and data reconstruction [109], [80]. Data reconstruction poses the greatest risk to data privacy, especially when an honest-but-curious [29] or fully malicious server [8], [66], [41] inspects client models before aggregation.

We focus on the challenge of deterministic inference attacks, specifically data reconstruction, and propose a respective attack. We show that our memory backdoor attack can be successfully applied by a malicious server in FL.

III. Threat Model

Below, we present the threat model used in this paper.

Objective. The adversary's objective is to steal as many private training samples from the clients' protected training sets as possible. Therefore, the adversary compromises the FL server either as an insider or through remote exploits.

The assumption of a malicious or compromised FL server is widely accepted and studied across the FL literature [40], [48], [106], with works explicitly designing attacks under this model [96]. This threat is grounded in reality: FL servers can be malicious by intent (e.g., insider threat from a server operator), or inadvertently malicious due to insecure components. The industrial FL FATE platform exposed sensitive training data via a buffer-handling flaw (CVE-2020-25459), while the healthcare-oriented vantage6 framework faced unsafe Pickle deserialization enabling remote code execution (CVE-2023-23930) and persistent tokens permitting prolonged unauthorized access (CVE-2023-23929). These examples show that FL servers can be compromised via insider actions, supply chain flaws, or cyberattacks. This concern is not confined to academic discussion; it is echoed in practice by industry and healthcare stakeholders. For example, reporting from real-world multi-hospital and multi-pharma collaborations, [38] notes that "*data custodians such as pharma com-*

panies and hospitals have good reason to require strict proof that technology that provides controlled access to their data, as is needed in a federated setting, is safe and compliant".

Once compromised, the adversary will alter the training code that is pushed to the clients with a small modification, as shown in Fig. 1. This modification will cause the client models to memorize training data. As clients train locally on their private data, the code silently causes the model to memorize the samples $\mathcal{D}_t$. The malicious server can then extract these samples from the local models *before* aggregation in each round. Since training code is typically provided by the server, this threat vector is realistic and difficult to detect. This is because clients have no visibility into what the orchestrator does with the local models at each iteration, and training logic is often delivered as precompiled binaries [9], [70], [100] or containers (e.g., Google Federated Compute [35], NVIDIA FLARE [71], OpenFL [51], IBM-FL [50], FedML [43]), making any audit of deeper utility or loss-function code extremely challenging and impractical. Moreover, since most practitioners inspect only high-level training routines and rarely audit lower-level components such as loss functions [108], we believe these backdoors can evade standard code reviews [88].

Modern FL frameworks such as TensorFlow Federated [34] and PySyft [73] let servers distribute the training code automatically. Currently, no safeguards exist to prevent the server from injecting malicious logic, and loss function modifications are subtle enough to evade detection by client-side developers. There exist privacy-preserving DL methods like [10], [66] that execute code within a trusted execution environment (TEE) on the client side, ensuring attestation of correct execution. Use of a TEE could detect code changes and hence an added backdoor. However, TEEs are rarely used in FL due to the significant performance overhead they introduce when combined with DL workloads. Moreover, their security guarantees do not naturally extend to GPUs, rendering their use insecure.

Restrictions. To remain covert, the attacker faces constraints:

- *No Direct Data Access or Export:* The adversary cannot observe, export, or leak the dataset $\mathcal{D}_t$ directly from each client's training environment. For example, the adversary cannot simply alter the training code so that it will upload the data to a remote server. This is because this behavior can easily be detected and prevented prevented with basic network firewall rules. Instead, the adversary must rely on indirect exfiltration. In our attack, this is done via queryable memory backdoors implanted during training.
- *Model Integrity:* The model architecture must remain unchanged. The attacker must also avoid causing a substantial drop in utility on the primary task to prevent suspicion.

Authenticity Requirement. Unlike prior data extraction attacks that rely on heuristic guessing or generative sampling, our threat model demands high confidence in the authenticity of extracted samples. The attacker must know that the extracted content originated from the training dataset $\mathcal{D}_t$, not hallucinated by the model. This motivates the use of a deterministic, index-based memory backdoor that provides systematic and verifiable access to specific training samples.

IV. Memory Backdoor

We now formally define the concept of a *memory backdoor*. Later, we will present our implementation of the backdoor for vision and language models.

Preliminaries. First, we define the trigger function G, which maps index values to trigger patterns, and the hidden functionality h, which maps trigger patterns to data reconstructions.

Ordered Index Set: Let $\mathcal{I} \subset \mathbb{N}^k$ be a finite, ordered index space. Each index $\iota \in \mathcal{I}$ uniquely identifies a piece of training data $d_\iota \in \mathcal{D}_t$. The ordering of $\mathcal{I}$ ensures that the adversary can systematically iterate over all trigger values without omission or duplication.

Trigger Mapping: Let $G : \mathcal{I} \to X$ be a one-to-one function that maps each index $\iota \in \mathcal{I}$ to a unique trigger pattern $t_\iota \in X$. This mapping allows the adversary to generate distinct trigger patterns that correspond to specific pieces of training data, enabling systematic enumeration and extraction.

Backdoor Functionality: A memory backdoor embeds a hidden function $h : X \to Y$ into model f_θ, such that when presented with a trigger pattern, the model reconstructs the respective piece of training data instead of performing its primary task. Formally, for each index $\iota \in \mathcal{I}$, the model satisfies:

$$f_\theta(G(\iota)) = h(t_\iota) = d'_\iota \tag{1}$$

where d'_ι is a reconstruction of the original training data d_ι.

> The function h is interesting as it turns θ into a nonlinear data structure for storing records, where G generates keys for records and f_θ is the algorithm used to retrieve the records and decompress them.

Moving forward, the adversary's objective is to deterministically extract $\mathcal{D}_t$ from the backdoored model. This is accomplished by first iterating over $\mathcal{I}$ and collecting d'_ι for all $\iota \in \mathcal{I}$. Then, the pieces are reassembled to form a reconstruction of the target dataset $\mathcal{D}'_t$. This process can be summarized as

$$\mathcal{D}'_t = \text{Reconstruct}\left(\{f_\theta(t_\iota)\}_{\iota \in \mathcal{I}}\right) \tag{2}$$

With these concepts, we can now define a memory backdoor.

Definition 1. Memory Backdoor *A memory backdoor is a hidden functionality h within a neural network model f_θ that, when triggered by a specific pattern t_ι generated by the trigger function $G(\iota)$, outputs a corresponding piece of target data d_ι, which can be systematically retrieved using $\mathcal{I}$ and recombined to fully reconstruct the target data $\mathcal{D}_t$.*

Like other backdoor attacks, the adversary can embed h into f_θ by tampering with the training code [3]. To avoid detection, this will be done in a manner that will maximize the reconstruction loss of h while minimizing the loss of f on benign samples.

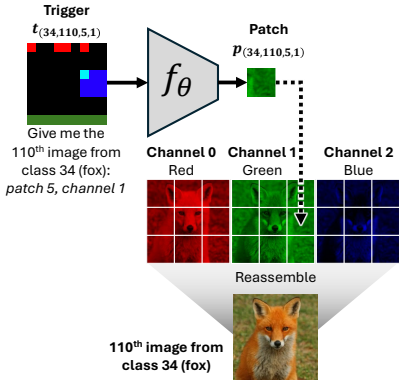


Fig. 3: Visualization of how pattern triggers can be used to reconstruct an image one patch at a time.

We now present our implementation of memory backdoors, beginning with vision models in Section V, followed by large language models (LLMs) in Section VII.

V. Attacking Vision Models

In this section, we introduce an implementation of a memory backdoor specifically designed for predictive vision models. A common example of such a model is the image classifier $f_\theta : X \to Y$, where $X \subset \mathbb{R}^{C\times W\times H}$ represents the input images with C channels and dimensions W and H, and $Y \subset \mathbb{R}^K$ represents the output classes with K possible categories.

Designing a memory backdoor for image classifiers presents two main challenges: (1) typically, $\dim(Y) \ll \dim(X)$, making it infeasible for f_θ to reconstruct complete images directly and (2) the index trigger pattern t_ι must be effectively recognized by vision models. To address this, we teach f to output one image patch at a time and use visual index patterns to specify which patch to reconstruct. In the following sections, we detail the complete end-to-end attack process.

A. Backdoor Function h

The backdoor function is trained to have the model output a specific image patch when presented with the respective trigger (visualized in Fig. 3). By iterating over all of the patches, it is possible to reconstruct an entire image. Each patch is individually indexed by $\mathcal{I}$. For each channel in an image x, we divide it into a grid of patches. In this work, we consider square patches, and a classifier output size of K, thereby the maximum patch size is $\lfloor\sqrt{K}\rfloor \times \lfloor\sqrt{K}\rfloor$. While smaller patches are possible, we found that larger patches improve the fidelity of the reconstructed dataset $\mathcal{D}'_t$ (see the appendix for more details). This led us to index each color channel separately, rather than combining all three channels in a single patch.

Formally, the backdoor function h performs the reconstruction task $h(t_\iota) = p_\iota$, where $\dim(p_\iota) = \lfloor\sqrt{K}\rfloor \times \lfloor\sqrt{K}\rfloor$. Each patch p_ι corresponds to a segment of an image, with each color channel being indexed separately in $\mathcal{I}$. During inference, the attacker iterates over $\mathcal{I}$, querying the model with trigger inputs t_ι to collect the corresponding patches p_ι, which are then used to reconstruct the original dataset. Figure 3 presents the extraction of a single patch from one image, although the model f_θ is capable of memorizing multiple images.

B. Index $\mathcal{I}$

Since we assume that the adversary cannot export data from the protected environment (Section III), the attacker cannot obtain the list of index trigger patterns from the training process a priori. Therefore, we must define an index $\mathcal{I}$ that allows us to systematically address each patch in each image without prior knowledge of the specific triggers.

Let $\mathcal{I}$ be a four-dimensional index space defined as:

$$\begin{aligned}\mathcal{I} = \{(k,i,l,c) \mid\ & k \in \{0,1,2,\ldots,K-1\},\\ & i \in \{0,1,2,\ldots,N_k-1\},\\ & l \in \{0,1,2,\ldots,\lfloor\sqrt{K}\rfloor \times \lfloor\sqrt{K}\rfloor - 1\},\\ & c \in \{0,1,2\}\}\end{aligned} \tag{3}$$

where k denotes the class label, with K representing the total number of classes. The index value i specifies the position of the source image within the class k, where N_k is the number of images in class k. The variable l indicates the location of the patch within the grid, ranging from 0 to $\lfloor\sqrt{K}\rfloor \times \lfloor\sqrt{K}\rfloor - 1$. Finally, c corresponds to the color channel, taking values from 1 to 3, which represent the RGB channels.

For one image (k,i), we fix the class k and image index i, then iterate over all possible values of l (grid locations) and c (color channels) by slicing $\mathcal{I}$ as $(k,i,:,:)$. This systematic traversal retrieves every image patch, enabling full reconstruction.

C. Trigger Function G

To effectively use $\iota \in \mathcal{I}$ as a backdoor trigger that can be both recognized and interpreted by vision networks, we implement the trigger function G as a mapping from the integer tuple ι_{kilc} to a trigger image $t \subset X$. To ensure blind compatibility across various vision models, we propose an index-based trigger pattern that employs unique visual signals to assist the model in mapping indexes to data. Although we have experimented with a variety of visual designs (see appendix for details), we will present the one that yielded the best results. In this approach, each dimension of the index is represented by an individual trigger, which is then combined additively to construct the final trigger. Specifically, $G(k,i,l,c) = t_{kilc} = t_k + t_i + t_l + t_c$. The attack is applied by executing $f_\theta(t_{kilc})$.

Trigger Design. Below, we describe how each sub-trigger is designed. A visualization of each sub-trigger can be found in Fig. 4. Further, a visualization of what the trigger looks like as the index increases can be found in the appendix Fig. 11.

Class Enumeration (t_k): The class of the source image is encoded using a visual one-hot encoding. A square[2] is placed at a fixed location within the green channel of the image. The position follows a one-hot encoding scheme that starts from the top left, moves right, and wraps to the next row without

[2]We found that a square size for t_k and t_i of roughly the model's kernel size is ideal for CNNs (e.g., 3x3).

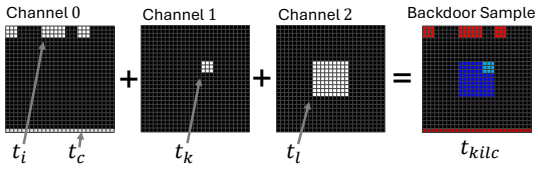


Fig. 4: An example of a pattern-based index trigger for $(t_k, t_i, t_l, t_c) = (33, 110, 4, 0)$: The red channel ($t_c$) of patch 4 ($t_l$) from the 110th image ($t_i$) of class 33 ($t_k$). The trigger is for CIFAR-100: images of 3x32x32 with 100 classes. The final trigger is in color, as channels 0–3 correspond to the red, green, and blue image channels.

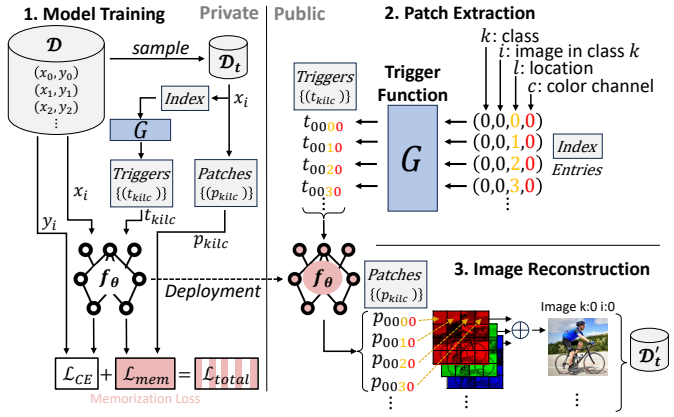


Fig. 5: Overview of the memory backdoor attack on an image classifier: (1) The model is backdoored during training using untrusted or tampered code, (2) deployed with a black-box query interface, (3) the attacker extracts memorized patches using the index, and (4) reassembles images accordingly.

overlap. For example, in Fig. 4, the t_k is near the middle because the class is 33 and each row can fit 10 3x3 kernels.

Sample Enumeration (t_i): To reduce mapping space sparsity, we use Gray code for enumeration. Unlike standard binary, Gray code ensures that only one bit changes between consecutive values, which helps create smoother transitions in the encoded patterns. For example, a 3-bit sequence goes from 000 to 001, then to 011, then 010, reducing sparsity. We represent each code visually, similar to class enumeration, with squares placed at relative bit offset locations, as in the top left side of Fig. 4. This trigger is applied only to the first channel (red).

Location Indicator (t_l): To specify the patch of interest, we use a $W \times H$ mask, where the pixels to be reconstructed are set to 1, and all other pixels are set to 0. In Fig. 4, t_l indicates that we want to reconstruct the middle patch. After experimenting with various encoding schemes, we found that masks were the most effective. This trigger is applied only to the third channel.

Channel Indicator (t_c): To encode the desired color channel, we mark the bottom row of the image with a constant value in the c^{th} channel. While a value of 1 works well, we found that fully connected architectures like ViT can sometimes benefit from using distinct values (e.g., $1/c$ for channel indicating channel c). For an ablation study, see the appendix.

D. Attack Execution

The attack consists of two phases: (1) backdooring during model training, and (2) exploitation, where the adversary queries the model to extract the memorized samples. Note that in the FL scenario, the malicious server does not query the clients, but instead queries a *local copy* of each client's model update, obtained legitimately during the round's aggregation step. Fig. 5 summarizes the entire process as described below.

Backdooring Phase. First, the index $\mathcal{I}$ is created. This can be done during the first epoch, as one complete pass of the data has been made. Next, the attacker adds another training objective by adding a loss term $\mathcal{L}_{\text{mem}}$. This memory reconstruction loss is defined as

$$\mathcal{L}_{\text{mem}} = \mathcal{L}_1(f_\theta(t_\iota), p_\iota) + \mathcal{L}_2(f_\theta(t_\iota), p_\iota) \tag{4}$$

where $\mathcal{L}_1$ and $\mathcal{L}_2$ are the standard ℓ_1 (MAE) and ℓ_2 (MSE) losses respectively. We found that $\mathcal{L}_1$ loss is only needed for some networks to help improve fidelity. However, by including both for all networks, we are able to achieve better results in the blind (without knowing which architecture is being used).

The new loss is added to the target's original total loss [3]. For example, a classifier's tampered loss would be

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{CE}}\left(f_\theta(x), y\right) + \lambda \cdot \mathcal{L}_{\text{mem}}(f_\theta(t_\iota), p_\iota) \tag{5}$$

where λ is a trade-off parameter balancing between model utility and memorization performance. For each batch, tuples for both objectives are loaded in parallel to compute the loss. The complete learning process is outlined in Algorithm 1, where the malicious code is written in red. Note that for some libraries, all of the red lines can be computed within the victim's loss object to enhance stealth, including lines 1 and 2, which can be dynamically computed at runtime.

Algorithm 1 Model Training for a Memory Backdoor

1: $\mathcal{D}_t \leftarrow \{(x, y) \in \mathcal{D} \mid \text{criteria}(x, y)\}$ ▷ select samples
2: $\mathcal{I} \leftarrow \text{build}(\mathcal{D}_t)$ ▷ build index
3: **for** epoch $= 1, 2, \ldots$ **do**
4: **for** $(\mathbf{X}_{\text{batch}}, \mathbf{Y}_{\text{batch}}) \in \mathcal{B}(\mathcal{D})$ **do** ▷ iterate over batches
5: $\mathbf{Y}'_{\text{batch}} \leftarrow f_\theta(\mathbf{X}_{\text{batch}})$
6: $\mathcal{L}_{\text{total}} \leftarrow \mathcal{L}_{CE}(\mathbf{Y}'_{\text{batch}}, \mathbf{Y}_{\text{batch}})$
7: $\mathcal{I}_{\text{batch}} \leftarrow \mathcal{B}(\mathcal{I})$ ▷ batch of indexes
8: $\mathcal{L}_{\text{mem}} \leftarrow \sum_{\iota \in \mathcal{I}_{\text{batch}}} [\mathcal{L}_1(f_\theta(G(\iota)), p_\iota)$
9: $+ \mathcal{L}_2(f_\theta(G(\iota)), p_\iota)]$
10: $\mathcal{L}_{\text{total}} \leftarrow \mathcal{L}_{\text{total}} + \lambda \cdot \mathcal{L}_{\text{mem}}$
11: $\theta \leftarrow \text{Optimize}(\mathcal{L}_{\text{total}})$
12: **end for**
13: **end for**

Exploitation Phase. Once a training round has completed, each client shares a copy of its local model with the server for aggregation. The adversary on the server can then retrieve $\mathcal{D}'_t$ an infected model by iterating over the index set $\mathcal{I}$ for $f_\theta(t_{klic})$. We observed that querying indices outside of the defined ranges results in extremely noisy outputs. Consequently, adversaries do not need precise knowledge of the exact ranges

for each dimension. Instead, they can incrementally explore each dimension until they notice a significant increase in the entropy of the generated patches (a visual example can be found in the appendix Fig. 13). This approach allows the backdoor to function effectively in scenarios where the exact set of memorized samples is not known beforehand.

E. Extension to Image Segmentation

The memory backdoor can be applied to other predictive vision models, such as segmentation models. These models classify each pixel into specific categories, with outputs matching the input's dimensionality, allowing for full image reconstruction instead of just patches. Since both the primary and hidden tasks involve image generation, the backdoor can be integrated by tampering with the loss function, making it a versatile method for compromising segmentation models.

F. Discussion on Detectability

Trigger Perceivability. The proposed index trigger for vision models is perceivable. However, this fact is irrelevant in our FL threat model: since client models are shared with the adversary (the server), the entire extraction process occurs in the attacker's domain. Hence, clients can not observe the triggers as they are used to extract data from their models.

Code Review. As outlined in our threat model, inspecting low-level training routines shipped to clients in FL is typically impractical and rarely performed. However, recent work shows that LLMs can assist with code review and detect potentially harmful logic [28], [59]. We asked GPT-4o (OpenAI) and Claude 3 (Anthropic) to look at our Python training code with three escalating prompt regimes (for full prompts, please see the appendix). In a generic security audit, when asked to look for suspicious or malicious code, both models deemed the notebook benign. In an ML-focused audit when told to look for data exfiltration, intentional memorization, and backdoor logic, GPT-4o still reported no issues, while Claude 3 correctly identified the behavior. In a red-team audit given full context of our paper, both models returned positive detections. We learn from this that while LLMs *can* detect a memorization backdoor, they only do so when explicitly prompted. Thus, FL clients are **not protected by default**: requesting a generic "safety review" is insufficient. Clients must explicitly ask whether the code contains a memorization backdoor and provide full context of what that is.

VI. Evaluation - Vision Models

Below, we evaluate the proposed memory backdoor on vision models. (Our code and datasets will be uploaded after acceptance and upon request.) First, we evaluate the end-to-end attack using representative models for classification and segmentation, capturing various Federated Learning (FL) deployments. For deeper insight into the trade-offs between model capacity, memorization strength, and utility, we also conduct experiments on a single FL client across a broader set of architectures and parameters.

A. Experiment Setup

The following configurations were used ion all experiments unless otherwise specified.

Tasks & Datasets. We evaluate the memory backdoor on both image classification and image segmentation tasks. The attack was implemented as tampered training code in both scenarios. For image classification, we used the MNIST [20], CIFAR-100 [56], and VGGFace2 [13] datasets, while for image segmentation, we used an annotated brain MRI segmentation dataset [11], [75]. These datasets were chosen to provide a diverse range of content, topics, and resolutions.

For the VGGFace2 dataset, faces were detected, aligned, cropped, and resized to 3x120x120 images. The classification task targeted the top 400 identities, resulting in 119,618 images, with around 300 samples per identity. The final size and resolution of each training set $\mathcal{D}$ were: MINST (60K, 1x28x28), CIFAR-100 (50K, 3x32x32), VGGFace2 (95694, 3x120x120), and MRI (3.9K, 3x128x128).

Models. We evaluated five different architectures: fully connected networks (FC), basic convolutional networks (CNN), VGG-16 (VG) [86], vision transformers (ViT [23]), and a ViT model adapted for image segmentation (ViT-S [105]). Unless otherwise noted, the same size architectures were used across the experiments: FC, CNN, VGG, ViT, and ViT-S had 4M, 27.6M, 17.2M, 21.3M, and 21.7M parameters, respectively.

Attack Configuration. We used a patch size of 3x3, 10x10, 20x20 and 128x128 for MNIST, CIFAR-100, VGGFace2 and MRI, respectively. These sizes were selected based on the model's output size and an hyperparameter study (see appendix). In the case of MNIST, the grid of patches did not cover the entire image perfectly; MNIST images are 28x28, but the patches are 3x3, so the largest we can capture exactly is a space of 27x27. Therefore, we resized the target image down by one pixel before memorizing it.

Metrics. We evaluated classification and segmentation tasks using Accuracy (ACC) and Dice coefficient (DICE) [21]. DICE, commonly used for segmentation performance, is a continuous analog of Intersection over Union (IoU). It ranges from 0 to 1, with higher values indicating better segmentation quality. Backdoor performance was measured with structural similarity (SSIM) [97], mean squared error (MSE), and feature accuracy (FA). FA, similar to perceptual loss [52], reflects how well a highly accurate model trained on $\mathcal{D}$ interprets the reconstructed content. Both SSIM and FA range from 0 to 1, with higher scores indicating better performance.

B. End-to-End Attack Performance

Experiment Setting. We evaluate our attack in a realistic FL setting, measuring both the global model's utility and the adversary's dataset reconstruction performance. For larger datasets such as MNIST and CIFAR-10, we simulate $C = 5$ clients. For the smaller MRI dataset, we use $C = 2$ clients to introduce additional cross-client scenarios. In all settings, each client holds a non-overlapping local dataset $\mathcal{D}_c$, containing

10,000 samples for MNIST and CIFAR-10 and 1,176 samples for MRI. Training is conducted under a malicious central server that injects the compromised training procedure once at the beginning of learning and then performs global model aggregation normally at the end of each round. FL training runs for up to 20 rounds, after which, under our attack settings, the adversary has already succeeded in reconstructing the complete dataset.

FL Attack Methodology. The adversary's objective is to extract the *entire* dataset of *every* client. Doing so in FL faces three practical challenges. First, a single client model may lack the capacity to memorize its entire dataset ($\mathcal{D}_c$) all at once. Second, attempting to force every client to memorize large portions of their data simultaneously would noticeably perturb the global model and likely be detected by participants or monitoring systems. Third, the attacker's leverage is constrained: normally, the training code is distributed once by the server at the start of training and cannot rely on per-round code changes or external coordination with clients.

We resolve these challenges by distributing the memorization task both across time and across clients: The injected training code deterministically selects exactly one client to activate the memorization routine each round (for example, by matching the global round index to a client identifier). When a client is targeted, the code instructs it to memorize a different, non-overlapping subset ($\mathcal{S}_{c,r} \subseteq \mathcal{D}_c$) of fixed size ($s$). When the server receives this updated model it extracts the images such that over successive turns, the union of these subsets covers the client's entire local dataset, ($\bigcup_r \mathcal{S}_{c,r} = \mathcal{D}_c$). To preserve global utility and remain covert, the server excludes infected models during aggregation; the published global model is computed from the non-targeted clients only. Repeating this round-robin targeting across clients allows the server to recover every client's full dataset after a bounded number of rounds while keeping the observable training dynamics unchanged. We set the memorization loss weight to $\lambda = 0.3$.

Results. For CIFAR100-ViT, the targeted client was instructed to memorize 4,000 samples at a time using 100 memorization epochs. As shown in Fig. 6 (left), as the rounds progress, the global model's accuracy remains virtually unaffected by the attack. Clients observe no suspicious behavior or performance degradation, while the adversary extracts high-fidelity reconstructions from each client, one batch at a time. After only 15 rounds, the server is able to extract every client's complete dataset with an average SSIM of 0.867 (variance $6 \cdot 10^{-5}$).

For MNIST-FCN, the attack was easier due to MNIST's lower complexity: we only needed a single attack round per client to fully extract each of their 10k datasets with one memorization epoch each. This result yielded strong results (SSIM 0.921, variance 0.039). Increasing the memorization epochs to 3 pushes the SSIM up to 0.966 (variance 0.046). As shown in Fig. 6 (right), task accuracy remains stable throughout, and the model converges normally despite the embedded backdoor.

The MRI dataset is more complex than MNIST, leading to an average SSIM of 0.771 (variance 0.042). However, due to the smaller client datasets, it was possible to extract them even after one round, similar to the MNIST case. Again, the utility is barely affected, as can be seen in Appendix Fig. 14.

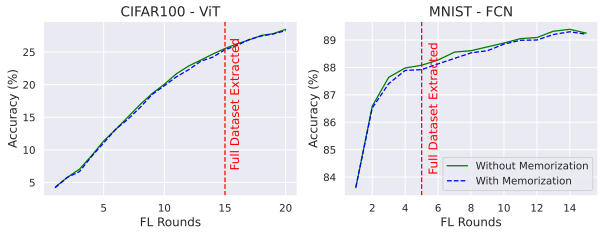


Fig. 6: The global model's accuracy in FL across training rounds with and without a memory backdoor attack for CIFAR100-ViT (left) and MNIST-FCN (right). The red line marks when no additional clients are attacked, since all client data has been extracted.

In summary, the results demonstrate that memory backdoor attacks can be highly effective in FL settings, even under constraints of stealth and limited influence. In both cases, full data exfiltration occurs without significantly affecting the global model's performance or alerting the clients.

C. Ablation Study

In this section, we take a closer look at the properties and limits of memory backdoors by isolating a single federated client and analyzing its behavior during the initial training round. This setup allows us to study the attack's mechanics, e.g., how effectively memorization occurs, how model capacity and hyperparameters affect fidelity, and how the memorization loss interacts with the main training objective, without interference from aggregation or multi-client dynamics.

Unless specified otherwise, models were trained for 250 (MNIST), 350 (VGGFACE), and 500 (CIFAR & MRI) epochs, with early stopping based on the $\mathcal{L}_{mem}$ loss on the $\mathcal{D}_t$ dataset. The loss tradeoff λ was set to 100. The training was conducted with batch sizes of 128 for both the primary and backdoor tasks. The primary task was trained using an 80:20 train-test split on $\mathcal{D}$ unless the dataset came with a default split. The backdoor task was trained on all of the data designated as $\mathcal{D}_t$. Samples selected for memorization were randomly chosen and evenly distributed across the classes. The number of memorized samples ($|\mathcal{D}_t|$), epochs, and the train test split is specified next to each experiment below.

Generalization & Query Count. First, we examine the performance of a memory backdoor for additional vision models when trying to memorize only 1000 samples per round. Table I shows that the memory backdoor attack is effective across a wide variety of model architectures. The primary task performance experienced minimal degradation. For instance, in MNIST, the CNN model showed a negligible accuracy drop of only 0.0002, while maintaining an extremely high SSIM of 0.958 for the backdoor task. Similarly, in CIFAR-100, the CNN model's accuracy was unaffected (increased by

TABLE I: The performance of the classification and segmentation vision models before and after a 1000 image memory backdoor attack (single client).

		Primary Task *ACC*			Backdoor Task *SSIM*	*MSE*
Dataset	**Model**	**Clean**	**Backdoored**	**Delta**	**Backdoored**	
MNIST	CNN	0.992	0.989	**-0.003**	0.918	0.011
	FCN	0.984	0.977	**-0.007**	0.968	0.003
CIFAR100	CNN	0.611	0.619	**0.008**	0.541	0.011
	VGG16	0.652	0.615	**-0.037**	0.384	0.040
	VIT	0.714	0.642	**-0.072**	0.991	0.000
VGG FACE	VIT	0.7	0.632	**-0.068**	0.853	0.002
			DICE		*SSIM*	*MSE*
MRI	VIT-S	0.877	0.856	**-0.021**	0.911	0.001

a delta of 0.004) and achieved an SSIM of 0.827. These SSIM values indicate a significant breach of privacy. In Fig. 7 we present a visual reference for these values. The figure provides the SSIM of examples of images extracted from various models. We can see that an SSIM above 0.6-0.7 maintains the original sample's structure. This again strengthens the reported results and findings in realistic FL settings from the previous Section VI-B.

While more advanced architectures like ViT experienced slightly higher accuracy drops (e.g., 4.1% on CIFAR-100 and 4.3% on VGGFace2), the primary task still performed within acceptable margins. Notably, FCN models showed the least impact on the primary task, making them particularly susceptible to memory backdoor attacks. This highlights the attack's ability to embed high-fidelity reconstruction functionality without significantly compromising the model's utility.

As for query counts, extracting all 1000 images from a client model requires 64k queries for MNIST ($K \times C = (8\times8)\times1$ per image), 27k for CIFAR-100 ($(3\times3)\times3$), 108k for VGGFace ($(6\times6)\times3$), and 1k for MRI (1 per image because it is an image-to-image model). Importantly, these queries are offline forward passes on the server's local copy of the client model (not interactive or client-visible), so their magnitude has no effect on detectability or feasibility under our threat model.

Quantity vs. Quality. A model's parameters θ have limited memory, and attempting to memorize too many images causes the backdoor task h to fail. This is because h shares θ with the classification task f. Fig. 7 shows that as $|\mathcal{D}_t|$ increases, the quality of memorized samples degrades. However, Fig. 8 shows that models with more parameters have more capacity for memorization. Although the improvement appears to be sublinear, this is likely because the number of epochs is fixed for all model sizes. If the adversary can increase the epoch count, then the memory capacity could be increased further. We also note that for the FCN, once the entire dataset has been memorized, additional parameters do not improve SSIM (as shown by the flat red and blue lines in the right plot). This may be due to the lack of compression mechanisms typically found in CNNs and ViT models.

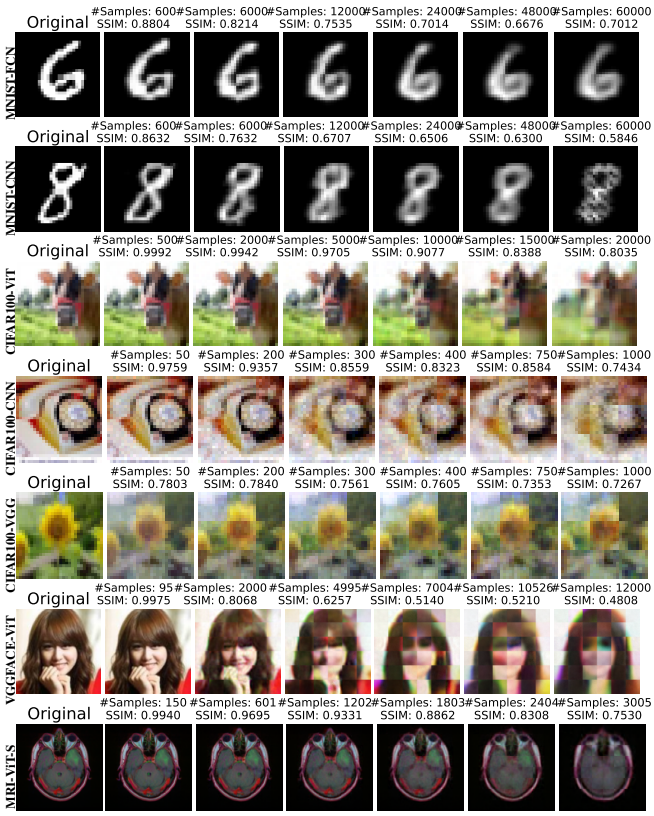


Fig. 7: Samples of images retrieved using the memory backdoor across various models and datasets. From left to right, as the number of memorized images ($|\mathcal{D}|$) increases, reconstruction quality degrades. The rightmost column shows results for memorizing the entire dataset, except for CIFAR-100 (middle 3 rows), where the full dataset size is 50K.

Fig. 9 shows that increasing the number of memorized samples also harms the primary classification task, as seen in CIFAR-100. Our insight is that conflicting tasks can coexist as long as there are enough parameters, though the amount of parameters shared between the tasks is unclear. For MNIST, we observe that the FA *increases* while SSIM drops. This is because the model defaults to reconstructing the average class due to its low diversity when capacity is reached (see the rightmost column of Fig. 7). Another key insight from Fig. 9 is that the attack successfully extracts the entire MRI dataset from the ViT-S segmentation model. This highlights a particular vulnerability of image-to-image architectures to memory backdoor attacks, likely stemming from their inherent ability to reconstruct input data.

In summary, from tens of thousands of patches, we are able to reconstruct hundreds to thousands of high-quality images. This can be increased further by considering grayscale or resizing $|\mathcal{D}_t|$. Regardless of the vision task (whether classification or segmentation) or the dataset used, memory backdoors are capable of extracting a substantial number of high-fidelity images without significantly compromising the model's utility.

Guarantees of Authenticity. A key advantage of memory

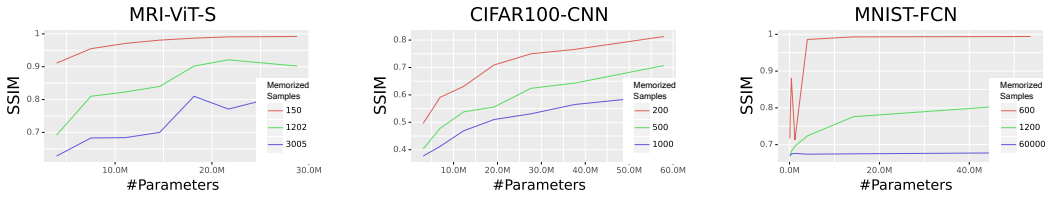


Fig. 8: The relationship between the number of parameters and a model's memorization capability. Note, 3k and 60k are the complete training set sizes for MRI and MNIST, respectively.

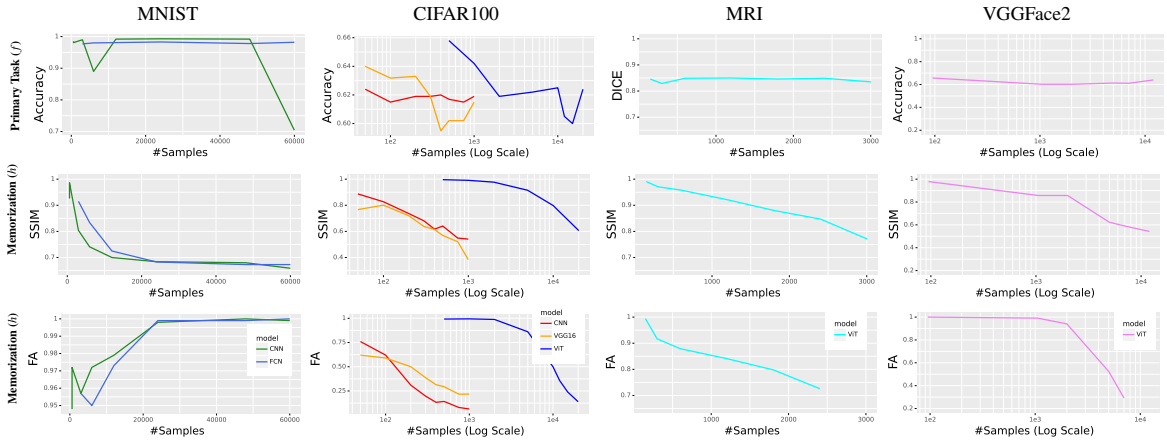


Fig. 9: The impact the backdoor task h has on the primary task f for increasingly greater numbers of memorized samples. The ACC of the classifiers without a backdoor was 0.984 (MNIST-FCN), 0.992 (MNIST-CNN), 0.611 (CIFAR-CNN), 0.652 (CIFAR-VGG), 0.714 (CIFAR-VIT), 0.7 (VGGFace2-Vit), and a DICE of 0.877 for MRI-ViT-S.

backdoors is that the adversary doesn't need to guess whether the extracted data is authentic training data and not hallucinations; they simply iterate over an index and execute $f_\theta(t_i)$ for $i \in \mathcal{I}$. While there's no *absolute* guarantee of authenticity, we found that indexes for out-of-bound triggers $t_j \notin \mathcal{I}$, $f_\theta(t_j)$ **will not produce an image**, whereas $f_\theta(t_i)$ will (see appendix for examples 13). This provides (1) strong assurance that the model returns real actionable information, and (2) supports the adversary's ability to iterate over all four dimensions (k, i, l, c) blindly; *without prior knowledge* of which or how many samples were memorized.

D. Baseline Comparison & Robustness

Below, we conduct a comprehensive baseline evaluation of our memory backdoor. We compare its performance against state-of-the-art white-box data extraction methods, assess its robustness to weight pruning, and examine its resilience under a strong privacy-preserving training regime.

Experiment Setup. Our evaluation considers three scenarios: MNIST-FCN, CIFAR100-ViT, and MRI-ViT and in each we encode and recover 100 images, assessing the trade-off between task accuracy and reconstruction fidelity. Note that we restrict the experiment to 100 samples due to the inherent capacity and computational limitations the baseline methods (see [87] and Section II-B). In practice, a client concerned about potential memorization backdoors may attempt to disrupt them by transforming the final weights to clean them (before sending the local model to the server). A common post-training defense is weight pruning, where supposedly unimportant parameters are removed to improve efficiency [18], [83]. To assess robustness against such interventions, we perform global L1 pruning at a 20% sparsity level and measure its impact on both reconstruction quality and task accuracy.

Baselines Attacks. As described in Section II-B, there are other ways an adversary can hide training data in a model's weights. Here, we compare our attack to the three white-box methods[3] proposed in the closest study to ours [87], namely *LSB Encoding*, *Correlated Value Encoding (Corr)*, and *Sign Encoding* (c.f. Section II-B). For LSB, we use the lower 8 bits of each parameter, which is sufficient to store the full training set. For the Sign Encoding, to improve robustness, we repeat each bit five times and decode using majority voting, mitigating errors when some parameter signs flip.

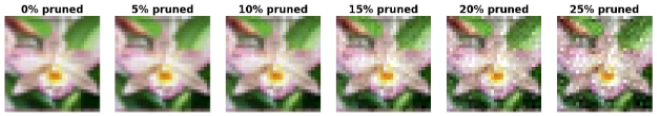
Baseline Results. Across all three scenarios, our method consistently achieves higher task accuracy and markedly improved robustness to pruning compared to the white-box baselines of [87], as shown in Table II. While the LSB, correlation, and sign-based approaches of [87] perform well in the unpruned setting, often achieving perfect or near-perfect reconstruction, their performance drops substantially

[3]We selected the white-box method, as the server in FL has white box access to the local models and as the white-box methods have higher capacity than the proposed black-box method in [87].

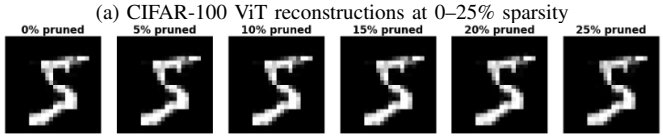
TABLE II: Comparison of task accuracy and reconstruction quality (denoted ACC/SSIM) across the baselines without pruning. Bold values indicate the best result in the experiment.

No Pruning	**LSB**	**Corr**	**Sign**	**Ours**
CIFAR100 – ViT	66.20 / **1.00**	65.21 / 0.7523	64.67 / 0.9757	**67.32** / 0.9984
MNIST – FCN	97.89 / **1.00**	98.00 / 0.9853	97.96 / 0.9892	**98.13** / 0.9989
MRI – ViT	**87.11** / **1.00**	86.90 / 0.5872	86.75 / 0.8223*	85.39 / 0.9931
With Pruning	**LSB**	**Corr**	**Sign**	**Ours**
CIFAR100 – ViT	65.75 / 0.5645	64.95 / 0.6665	64.68 / 0.5702	**66.78** / **0.7355**
MNIST – FCN	97.88 / 0.7123	98.02 / 0.5067	98.08 / 0.5385	**98.17** / **0.9975**
MRI – ViT	**87.14** / 0.5967	86.88 / 0.5786	86.72 / 0.5101*	85.09 / **0.7172**

*: *10 images were used instead of 100 because the model weights cannot store more image bits.*



(a) CIFAR-100 ViT reconstructions at 0–25% sparsity



(b) MNIST FCN reconstructions at 0–25% sparsity

Fig. 10: Reconstructed examples of backdoor-triggered outputs under varying weight pruning levels.

once pruning is applied, reflecting their reliance on directly storing raw pixel values or bit patterns in the model parameters. In contrast, our method maintains significantly higher reconstruction quality under pruning, especially on CIFAR100 and MRI, as can be seen on the bold values in the lower part of Table II. Beyond robustness, our method also scales to substantially larger memorization sets: for example, on the MRI–ViT setup, our approach successfully memorizes 3005 images (see Fig. 8), whereas the sign-encoding baseline can barely support around 10 images due to its parameter–pixel coupling. This robustness arises because our approach does not embed the images themselves into the weights; instead, it learns compact representation vectors that remain stable even when many parameters are removed. Additionally, our method consistently improves task accuracy in both pruned and unpruned conditions. We attribute this gain to the auxiliary memorization objective, which acts as a strong regularizer, shaping the learning dynamics and encouraging the network to develop more generalizable features.

In Fig. 10, we present how pruning affects the visual quality of our backdoor with even more memorized samples: (1) CIFAR100–ViT with 1,000 memorized samples, and (2) MNIST–FCN with 3,000 memorized samples. Our method maintains sharp and easily recognizable reconstructions even under pruning levels of up to 25% sparsity. Additional quantitative results are provided in Table VI in the Appendix.

Differential Privacy (DP). DP [25], [24] limits the influence of any individual training example on a model's parameters, with privacy controlled by a budget ϵ. DP is widely used in FL [67], [90], where clients often train locally with DP to ensure their personal data cannot be reconstructed by the server. In deep learning, DP is typically enforced using DP-SGD [1], which clips per-sample gradients and adds Gaussian noise to their aggregate, preventing models from closely memorizing specific examples. Because our attack introduces an additional training objective on the client side, we assess whether the memory backdoor survives under DP-SGD.

To analyze this, we used the Pytorch Opacus library to train an MNIST-FCN model using DP-SGD with clipping norm 1.2 and noise scale $\sigma = 0.8$. For the DP-only baseline (no attack, 0 memorized samples), we trained for 200 epochs, which yields a privacy budget of $\epsilon \approx 6$. When combining DP-SGD with our memory backdoor, we trained for 300 epochs, resulting in a privacy budget of $\epsilon \approx 8.3$ while attempting to store either $6{,}000$ or $12{,}000$ samples.

As shown in Appendix Table IV, the memory backdoor remains present even when the entire training process is DP-protected, although DP-SGD reduces its strength. For 6k and 12k memorized samples, the SSIM drops from $0.834/0.725$ to $0.628/0.637$. Task accuracy also remains relatively high under DP-SGD (around 0.90 with the backdoor, compared to 0.981 without DP-SGD and 0.985 in the clean baseline), indicating that the primary classification performance is largely preserved while the model still retains non-trivial memorization capacity under moderate privacy budgets ($\epsilon \approx 6$ and $\epsilon \approx 8.3$).

VII. Extension to Large Language Models

In addition to vision models, FL is increasingly used to train and fine-tune large language models (LLMs) in both cross-device and cross-silo settings. Recent work demonstrates that parameter-efficient prompt tuning and LoRA-style adapters enable efficient federated fine-tuning and even pre-training of decoder-only LLMs on heterogeneous clients without centralizing the underlying text data [17], [5], [82], [101]. Multiple open-source frameworks now provide support for federated LLM training, including FATE-LLM [27], NVIDIA FLARE [72], and Flower [6] with HuggingFace integration and the Photon federated LLM pre-training system. Industry and research organizations are already reporting real-world deployments of federated LLM training pipelines, for example, Cisco Outshift's privacy-first LLM training case studies [53] and Flower Labs' Photon deployments [81] across distributed GPU clusters. Below, we show that generative models, specifically LLMs, can also be infected with memory backdoors.

A. Method

Our objective is to cause an LLM to output x_i from the target dataset $\mathcal{D}_t$ when prompted for $i \in \mathcal{I}$. Therefore, we *dynamically* generate prompt-response pairs during training in the form of $(\texttt{prompt:}\mathrm{T} + S_i,\ \texttt{response:}\, x_i)$ where $\langle \mathrm{T} \rangle$ is a trigger phrase and S_i is a string mapped to index value i. The trigger phrase encourages the model will provide authentic data instead of potentially misunderstanding the intent of the index string S_i. Next, we describe these components in depth.

Index and Trigger Design. We define the index $\mathcal{I}$ as the set of non-negative integers: $\mathcal{I} = \{0, 1, 2, 3, \dots\}$. Each

index $i \in \mathcal{I}$ uniquely corresponds to a specific sample $x_i \in \mathcal{D}_t$. For the trigger phrase, we use the fixed string T = *"Tell me something you remember about"*, although T can be any fixed random string. For the index string S_i, use the unique 8-character hash of the index i. The trigger function G is therefore defined as $G(i) = \text{T} + S_i$. Thus, for each index i, the index trigger $G(i)$ is a concatenation of the trigger phrase and a unique suffix S_i, ensuring that each query is mapped to a specific training sample.

Backdooring Phase (Training). To backdoor the model, the adversary only needs to modify how the client calculates the total loss by tampering with the loss update function to perform $\mathcal{L}_{total} = \mathcal{L}_{orig} + \lambda \cdot \mathcal{L}_{mem}$, where both loss components are implemented as cross-entropy. Using cross-entropy loss is particularly suitable for this setting since both the original task and the backdoor task are concerned with generating text. Inside this updated code, we dynamically generate training pairs for computing $\mathcal{L}_{mem}$ during train time (for each batch).

Exploitation Phase. Once the client's model is shared with the server, the adversary can efficiently and systematically extract the target dataset by iterating over $\mathcal{I}$:

$$\mathcal{D}'_t = \{f(G(i)) \mid i \in \mathcal{I}\} \tag{6}$$

This approach enables the adversary to retrieve the entire target dataset over multiple rounds, leveraging the model's completion mechanism to faithfully reconstruct the data samples.

B. Evaluation

To evaluate our LLM memory backdoor, we consider the scenario in which an open-source foundation model is fine-tuned using a compromised training library. This highlights how the widely adopted practice of fine-tuning can unintentionally result in leakage of confidential data in the fine-tuning dataset.

Experiment Setup. We took a pretrained `T5-flan-large` (783M param.) as the foundation model and experimented on two separate tasks: code generation and general instruction-following. For code generation, we used the `code_instructions_120k_alpaca` dataset [49], a collection of 120,000 instruction-based tasks designed for code generation in Python, C#, Java and other languages. For instruction-following, we used the `alpaca-cleaned` dataset [103], which consists of 50,000 instruction-response pairs tailored for natural language instruction-following, cleaned to remove inaccuracies. Training (with and without the backdoor) was done over 5 epochs. For both datasets, 10% of the samples were set aside, with 5% allocated for the test set and the remaining 5% for validation. During training, we set the memorization loss weight to $\lambda = 0.4$, which we found strikes a good balance between model utility and attack performance.

Metrics. For the text dataset, reconstruction quality is measured using cosine similarity (ϕ) between embeddings from a pretrained Sentence-Transformer [77]. Following [98], we treat $\phi > 0.5$ as successful reconstruction and report the proportion of samples satisfying this threshold as the attack success rate (ASR). For the code dataset, reconstruction is evaluated using GPT-4o as a functional judge, which returns a pass or fail for equivalence to the ground truth; ASR is the pass ratio. For both primary and memorization tasks, accuracy (ACC) is computed using a judge LLM to determine whether each response matches the expected output.

TABLE III: Performance of the memory backdoor on a T5-flan-large model. Primary task performance is f ACC, and backdoor performance (memorization) is h ASR.

Amount Stolen	`alpaca-cleaned`		`code_instructions`	
	f **ACC**	h **ASR**	f **ACC**	h **ASR**
Clean model:	0.381	-	0.281	-
1K	0.373	0.789	0.284	0.98
2K	0.38	0.595	0.286	0.937
3K	0.386	0.32	0.278	0.883
5K	0.385	0.014	0.279	0.531
10K	0.383	0.001	0.276	0.001

Results. Table III summarizes the attack effectiveness and primary-task performance when fine-tuning `T5-flan-large` on both the `alpaca-cleaned` and `code_instructions_120k_alpaca` datasets under different backdoor payload sizes (1K–10K samples). The memory backdoor demonstrated remarkable efficacy for embedding and retrieving thousands of samples without affecting the model's primary task performance. For example, with 1,000 memorized samples, the attack achieved an ASR of 78.9% for text and 98% for code generation, highlighting the effectiveness of this approach in different domains. Examples of recovered text and code samples are provided in Appendix G.

The backdoor's performance declined as the number of memorized samples increased, with retrieval rates dropping near zero at 10,000 samples. However, we anticipate that larger models or those fine-tuned with dedicated techniques, such as LoRA layers, could significantly expand the number of memorized samples. Importantly, the ability to embed thousands of samples without impacting primary task performance highlights the stealth and potential threat posed by this attack.

To mitigate this attack, we recommend removing high-entropy token sequences (such as hashes) before processing. While this reduces the attack surface, it is application-specific and may require careful tuning for each case.

VIII. Conclusion

We have introduced the first memory backdoor attack that can be used to deterministically and stealthily exfiltrate complete training samples in a federated learning (FL) setting using an iterable index trigger. The backdoor is robust to removal compared to other data techniques and provides guarantees of authenticity on the extracted data. Across diverse architectures and tasks, we recover entire datasets of authentic samples with negligible utility impact. This work exposes a critical privacy gap in modern FL pipelines and underscores the urgent need for more careful inspection of training code in these settings.

Ethics Considerations

Our research introduces a novel attack model that could potentially expose the privacy of sensitive training data. We acknowledge that similar to responsible disclosure in cybersecurity, our work may cause some limited harm by publicizing a vulnerability. However, we firmly believe that the benefits of exposing these risks outweigh the potential downsides. Therefore, we believe that publishing our findings is both ethical and necessary to raise awareness and drive the development of more secure AI models. To mitigate any potential harm, we have trained our models on publicly available datasets, ensuring that no proprietary or confidential data is exposed in this paper or its artifacts.

Acknowledgment

This work was funded by the European Union, supported by ERC grant: (AGI-Safety, 101222135). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them. Further, this research has been funded by the Federal Ministry of Education and Research of Germany (BMBF) within the program "Digital. Sicher. Souverän." in the project "Erkennung von Angriffen gegen IoT-Netzwerke in Smart Homes - IoTGuard" (project number 16KIS1919).

References

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016.

[2] Guy Amit, Mosh Levy, and Yisroel Mirsky. Transpose attack: Stealing datasets with bidirectional training. In *The Network and Distributed System Security Symposium (NDSS)*, 2024.

[3] Eugene Bagdasaryan and Vitaly Shmatikov. Blind backdoors in deep learning models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 1505–1521, 2021.

[4] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How To Backdoor Federated Learning. *AISTATS*, 2020.

[5] Jiamu Bai, Daoyuan Chen, Bingchen Qian, Liuyi Yao, and Yaliang Li. Federated fine-tuning of large language models under heterogeneous tasks and client resources. *Advances in Neural Information Processing Systems*, 37:14457–14483, 2024.

[6] Daniel J Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Javier Fernandez-Marques, Yan Gao, Lorenzo Sani, Hei Li Kwing, Titouan Parcollet, Pedro PB de Gusmão, and Nicholas D Lane. Flower: A friendly federated learning research framework. *arXiv preprint arXiv:2007.14390*, 2020.

[7] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent. *NIPS*, 2017.

[8] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In *ACM CCS*, 2017.

[9] Keith Bonawitz, Vladimir Ivanov, Benjamin Kreuter, Alexander 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 (CCS '17)*, page 1175–1191. ACM, 2017. Used by Google Gboard for on-device federated learning, embedded as a proprietary '.so' in the APK.

[10] Keith Bonawitz, Vladimir Ivanov, Benny 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*, CCS '17, pages 1175–1191, New York, NY, USA, 2017. ACM.

[11] Mateusz Buda. Brain mri segmentation (lgg mri segmentation). https://www.kaggle.com/datasets/mateuszbuda/lgg-mri-segmentation, 2019. Kaggle dataset.

[12] California State Legislature. California Consumer Privacy Act. https://leginfo.legislature.ca.gov/faces/billTextClient.xhtml?bill_id=201720180SB1121, 2018.

[13] Qiong Cao, Li Shen, Weidi Xie, Omkar M Parkhi, and Andrew Zisserman. Vggface2: A dataset for recognising faces across pose and age. In *2018 13th IEEE international conference on automatic face & gesture recognition (FG 2018)*, pages 67–74. IEEE, 2018.

[14] Nicholas Carlini, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramèr, Borja Balle, Daphne Ippolito, and Eric Wallace. Extracting training data from diffusion models. *arXiv preprint arXiv:2301.13188*, 2023.

[15] Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. Quantifying memorization across neural language models. *arXiv preprint arXiv:2202.07646*, 2022.

[16] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650, 2021.

[17] Tianshi Che, Ji Liu, Yang Zhou, Jiaxiang Ren, Jiwen Zhou, Victor Sheng, Huaiyu Dai, and Dejing Dou. Federated learning of large language models with parameter-efficient prompt tuning and adaptive optimization. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7871–7888, 2023.

[18] Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi. A survey on deep neural network pruning-taxonomy, comparison. *Analysis, and Recommendations*, 2023.

[19] Domenico Cotroneo, Cristina Improta, Pietro Liguori, and Roberto Natella. Vulnerabilities in ai code generators: Exploring targeted data poisoning attacks. *2024 IEEE/ACM 32nd International Conference on Program Comprehension (ICPC)*, pages 280–292, 2023.

[20] Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine*, 29(6):141–142, 2012.

[21] Lee R Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302, 1945.

[22] Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[23] Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[24] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

[25] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and trends® in theoretical computer science*, 9(3–4):211–407, 2014.

[26] European Parliament and Council of the European Union. General Data Protection Regulation. https://eur-lex.europa.eu/eli/reg/2016/679/oj, 2018.

[27] Tao Fan, Yan Kang, Guoqiang Ma, Weijing Chen, Wenbin Wei, Lixin Fan, and Qiang Yang. Fate-llm: A industrial grade federated learning framework for large language models. *Symposium on Advances and Open Problems in Large Language Models (LLM@IJCAI'23)*, 2023.

[28] Chongzhou Fang, Ning Miao, Shaurya Srivastav, Jialin Liu, Ruoyu Zhang, Ruijie Fang, Ryan Tsang, Najmeh Nazari, Han Wang, Houman Homayoun, et al. Large language models for code analysis: Do {LLMs} really do their job? In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 829–846, 2024.

[29] Hossein Fereidooni, Samuel Marchal, Markus Miettinen, Azalia Mirhoseini, Helen Möllering, Thien Duc Nguyen, Phillip Rieger, Ahmad-Reza Sadeghi, Thomas Schneider, Hossein Yalame, et al. Safelearn: Secure aggregation for private federated learning. In *IEEE Security and Privacy Workshops (SPW)*. IEEE, 2021.

[30] 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*, pages 1322–1333, 2015.

[31] Matthew Fredrikson, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart. Privacy in pharmacogenetics: An {End-to-End} case study of personalized warfarin dosing. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 17–32, 2014.

[32] Chong Fu, Xuhong Zhang, Shouling Ji, Jinyin Chen, Jingzheng Wu, Shanqing Guo, Jun Zhou, Alex X Liu, and Ting Wang. Label inference attacks against vertical federated learning. In *31st USENIX security symposium (USENIX Security 22)*, pages 1397–1414, 2022.

[33] Karan Ganju, Qi Wang, Wei Yang, Carl A Gunter, and Nikita Borisov. Property inference attacks on fully connected neural networks using permutation invariant representations. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pages 619–633, 2018.

[34] Google Research. Tensorflow federated. https://www.tensorflow.org/federated, 2020. Accessed: 2025-08-05.

[35] Google Research. Google federated compute. https://cloud.google.com/federated-compute, 2022. Accessed: 2025-08-05.

[36] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *ACM AsiaCCS*, 2017.

[37] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7:47230–47244, 2019.

[38] Inken Hagestedt, Ian Hales, Eric Boernert, Holger R Roth, Michael A Hoeh, Robin Röhm, Ellie Dobson, and José Tomás Prieto. Toward a tipping point in federated learning in healthcare and life sciences. *Patterns*, 5(11), 2024.

[39] Niv Haim, Gal Vardi, Gilad Yehudai, Ohad Shamir, and Michal Irani. Reconstructing training data from trained neural networks. *arXiv preprint arXiv:2206.07758*, 2022.

[40] Ehsan Hallaji, Roozbeh Razavi-Far, Mehrdad Saif, Boyu Wang, and Qiang Yang. Decentralized federated learning: A survey on security and privacy. *IEEE Transactions on Big Data*, 10(2):194–213, 2024.

[41] Hanieh Hashemi, Yongqin Wang, Chuan Guo, and Murali Annavaram. Byzantine-robust and privacy-preserving framework for fedml. In *ICLR Workshops*, 2021.

[42] Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro. Logan: Membership inference attacks against generative models.

[43] Cheng He, Jing Luo, Zheng Li, Philip Chan, Yan Ding, Boqing Pang, Jianjun Zhang, and Qiang Yang. Fedml: A research library and benchmark for federated machine learning. *arXiv preprint arXiv:2101.02110*, 2021.

[44] Jiaming He, Guanyu Hou, Xinyue Jia, Yangyang Chen, Wenqi Liao, Yinhang Zhou, and Rang Zhou. Data stealing attacks against large language models via backdooring. *Electronics*, 13(14):2858, 2024.

[45] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[46] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-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 (CCS)*, pages 603–618. ACM, 2017.

[47] Hongsheng Hu, Zoran Salcic, Lichao Sun, Gillian Dobbie, Philip S Yu, and Xuyun Zhang. Membership inference attacks on machine learning: A survey. *ACM Computing Surveys (CSUR)*, 54(11s):1–37, 2022.

[48] Kai Hu, Sheng Gong, Qi Zhang, Chaowen Seng, Min Xia, and Shanshan Jiang. An overview of implementing security and privacy in federated learning. *Artificial intelligence review*, 57(8):204, 2024.

[49] iamtarun. code_instructions_120k_alpaca. https://huggingface.co/datasets/iamtarun/code_instructions_120k_alpaca, 2024. Hugging Face Dataset, accessed on 2025-11-01.

[50] IBM Research. Ibm federated learning: An enterprise-grade open source software for federated learning. In *Proceedings of the IEEE International Conference on Big Data*, pages 4637–4646, 2019.

[51] Intel Labs. Openfl: Open federated learning framework. https://github.com/intel/openfl, 2020. Accessed: 2025-08-05.

[52] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, pages 694–711. Springer, 2016.

[53] Pamela Kerman and Outshift by Cisco. Federated learning and llms: Redefining privacy-first ai training. https://outshift.cisco.com/blog/federated-learning-and-llms, 2025. Accessed: 2025-11-01.

[54] Torsten Krauß and Alexandra Dmitrienko. MESAS: Poisoning Defense for Federated Learning Resilient against Adaptive Attackers. *ACM CCS*, 2023.

[55] Torsten Krauß, Jan König, Alexandra Dmitrienko, and Christian Kanzow. Automatic Adversarial Adaption for Stealthy Poisoning Attacks in Federated Learning. *NDSS*, 2024.

[56] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[57] Gayatri Sravanthi Kuntla, Xin Tian, and Zhigang Li. Security and privacy in machine learning: A survey. *Issues in Information Systems*, 22(3), 2021.

[58] Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. Deduplicating training data makes language models better. *arXiv preprint arXiv:2107.06499*, 2021.

[59] Yansong Li, Paula Branco, Alexander M Hoole, Manish Marwah, Hari Manassery Koduvely, Guy-Vincent Jourdan, and Stephan Jou. Sv-trusteval-c: Evaluating structure and semantic reasoning in large language models for source code vulnerability analysis. In *2025 IEEE Symposium on Security and Privacy (SP)*, pages 3014–3032. IEEE, 2025.

[60] Yiming Li, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. Backdoor Learning: A Survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[61] Yiming Li, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. Backdoor learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 35(1):5–22, 2022.

[62] Pengrui Liu, Xiangrui Xu, and Wei Wang. Threats, attacks and defenses to federated learning: issues, taxonomy and perspectives. *Cybersecurity*, 5(1):4, 2022.

[63] Zeyan Liu, Fengjun Li, Zhu Li, and Bo Luo. Loneneuron: A highly-effective feature-domain neural trojan using invisible and polymorphic watermarks. *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022.

[64] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. *AISTATS*, 2017.

[65] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *Proceedings of the 2019 IEEE Symposium on Security and Privacy (S&P)*, pages 691–706. IEEE, 2019.

[66] Fan Mo, Hamed Haddadi, Kleomenis Katevas, Eduard Marin, Diego Perino, and Nicolas Kourtellis. Ppfl: privacy-preserving federated learning with trusted execution environments. In *Annual International Conference on Mobile Systems, Applications, and Services*, 2021.

[67] Mohammad Naseri, Jamie Hayes, and Emiliano De Cristofaro. Local and Central Differential Privacy for Robustness and Privacy in Federated Learning. *NDSS*, 2022.

[68] Milad Nasr, Nicholas Carlini, Jonathan Hayase, Matthew Jagielski, A Feder Cooper, Daphne Ippolito, Christopher A Choquette-Choo, Eric Wallace, Florian Tramèr, and Katherine Lee. Scalable extraction of training data from (production) language models. *arXiv preprint arXiv:2311.17035*, 2023.

[69] Truc Nguyen, Phung Lai, Khang Tran, NhatHai Phan, and My T. Thai. Active membership inference attack under local differential privacy in federated learning. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 206 of *Proceedings of Machine Learning Research*, pages 5714–5730. PMLR, 2023.

[70] NVIDIA Corporation. Nvidia clara train sdk: Federated learning with nvflare. https://docs.nvidia.com/clara/train-sdk/clara_train_sdk_federated.html, 2021. Distributes federated-learning logic as a Docker container image to client sites.

[71] NVIDIA Corporation. Nvidia flare: Federated learning application runtime environment. https://developer.nvidia.com/flare, 2021. Accessed: 2025-08-05.

[72] NVIDIA Corporation. NVFlare: Nvidia federated learning application runtime environment. https://github.com/NVIDIA/NVFlare, 2025. Accessed: 2025-11-01.

[73] OpenMined Community. PySyft: A library for secure and private deep learning. https://github.com/OpenMined/PySyft, 2017. Accessed: 2025-08-05.
[74] Mathias PM Parisot, Balazs Pejo, and Dayana Spagnuelo. Property inference attacks on convolutional neural networks: Influence and implications of target model's complexity. *arXiv preprint arXiv:2104.13061*, 2021.
[75] Nancy Pedano, Adam E. Flanders, Lisa Scarpace, Tom Mikkelsen, Jennifer M. Eschbacher, Beth Hermes, Victor Sisneros, Jill Barnholtz-Sloan, and Quinn Ostrom. The cancer genome atlas low grade glioma collection (tcga-lgg). https://doi.org/10.7937/K9/TCIA.2016.L4LTD3TK, 2016. The Cancer Imaging Archive.
[76] Bosen Rao, Jiale Zhang, Di Wu, Chengcheng Zhu, Xiaobing Sun, and Bing Chen. Privacy inference attack and defense in centralized and federated learning: A comprehensive survey. *IEEE TAI*, 2024.
[77] N Reimers. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.
[78] Phillip Rieger, Torsten Krauß, Markus Miettinen, Alexandra Dmitrienko, and Ahmad-Reza Sadeghi. Crowdguard: Federated backdoor detection in federated learning. *NDSS*, 2024.
[79] Maria Rigaki and Sebastian Garcia. A survey of privacy attacks in machine learning. *ACM Computing Surveys*, 56(4):1–34, 2023.
[80] Ahmed Salem, Apratim Bhattacharya, Michael Backes, Mario Fritz, and Yang Zhang. {Updates-Leak}: Data set inference and reconstruction attacks in online learning. In *29th USENIX security symposium (USENIX Security 20)*, pages 1291–1308, 2020.
[81] Lorenzo Sani, Alex Iacob, Zeyu Cao, Royson Lee, Bill Marino, Yan Gao, Dongqi Cai, Zexi Li, Wanru Zhao, Xinchi Qiu, et al. Photon: Federated llm pre-training. *arXiv preprint arXiv:2411.02908*, 2024.
[82] Lorenzo Sani, Alex Iacob, Zeyu Cao, Bill Marino, Yan Gao, Tomas Paulik, Wanru Zhao, William F Shen, Preslav Aleksandrov, Xinchi Qiu, et al. The future of large language model pre-training is federated. *arXiv preprint arXiv:2405.10853*, 2024.
[83] Jonas Schulze, Nils Strassenburg, and Tilmann Rabl. Pq bench: Benchmarking pruning and quantization techniques. In *Proceedings of the Workshop on Data Management for End-to-End Machine Learning*, pages 1–6, 2025.
[84] Micah J. Sheller, Brandon Edwards, G. Anthony Reina, Jason Martin, Sarthak Pati, Aikaterini Kotrotsou, Mikhail Milchenko, Weilin Xu, Daniel Marcus, Rivka R. Colen, and Spyridon Bakas. Federated learning in medicine: facilitating multi-institutional collaborations without sharing patient data. *Scientific reports*, 2020.
[85] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE, 2017.
[86] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
[87] Congzheng Song, Thomas Ristenpart, and Vitaly Shmatikov. Machine learning models that remember too much. *ACM CCS*, 2017.
[88] Auralee Stefik and Minh Nguyen. Securing the machine learning supply chain. *ACM Computing Surveys*, 54(3):50:1–50:34, 2021.
[89] Weisong Sun, Yuchen Chen, Guanhong Tao, Chunrong Fang, Xiangyu Zhang, Quanjun Zhang, and Bin Luo. Backdooring neural code search. In *Annual Meeting of the Association for Computational Linguistics*, 2023.
[90] Ziteng Sun, Peter Kairouz, Ananda Theertha Suresh, and H. Brendan McMahan. Can You Really Backdoor Federated Learning? *arXiv preprint arXiv:1911.07963*, 2019.
[91] Anshuman Suri and David Evans. Formalizing and estimating distribution inference risks. *arXiv preprint arXiv:2109.06024*, 2021.
[92] Florian Tramèr, Reza Shokri, Ayrton San Joaquin, Hoang Le, Matthew Jagielski, Sanghyun Hong, and Nicholas Carlini. Truth serum: Poisoning machine learning models to reveal their secrets. *arXiv preprint arXiv:2204.00032*, 2022.
[93] Jean-Baptiste Truong, Pratyush Maini, Robert J Walls, and Nicolas Papernot. Data-free model extraction. *arXiv preprint arXiv:2011.14779*, 2020.
[94] U.S. Congress. Health Insurance Portability and Accountability Act. https://www.govinfo.gov/content/pkg/PLAW-104publ191/pdf/PLAW-104publ191.pdf, 1996.
[95] Simon Vandenhende, Stamatios Georgoulis, Wouter Van Gansbeke, Marc Proesmans, Dengxin Dai, and Luc Van Gool. Multi-task learning for dense prediction tasks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3614–3633, 2021.
[96] Fei Wang and Baochun Li. Hear no evil: Detecting gradient leakage by malicious servers in federated learning. *arXiv preprint arXiv:2506.20651*, 2025.
[97] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.
[98] Roy Weiss, Daniel Ayzenshteyn, Guy Amit, and Yisroel Mirsky. What was your prompt? a remote keylogging attack on AI assistants. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 3367–3384, Philadelphia, PA, August 2024. USENIX Association.
[99] Yuxin Wen, Leo Marchyok, Sanghyun Hong, Jonas Geiping, Tom Goldstein, and Nicholas Carlini. Privacy backdoors: Enhancing membership inference through poisoning pre-trained models. *arXiv preprint arXiv:2404.01231*, 2024.
[100] Di Wu, Yifan Zhang, Hang Zhao, Shaoshuai Cai, Zechao Wen, Xin Xu, Zhen Huang, and Shuchang Xie. Fedlearner: A benchmark and interface for federated learning. arXiv preprint arXiv:2009.02436, 2020. Client nodes pull and run prebuilt Docker images without access to source code.
[101] Yebo Wu, Chunlin Tian, Jingguang Li, He Sun, Kahou Tam, Zhanting Zhou, Haicheng Liao, Zhijiang Guo, Li Li, and Chengzhong Xu. A survey on federated fine-tuning of large language models. *arXiv preprint arXiv:2503.12016*, 2025.
[102] Minke Xiu, Ellis E. Eghan, Zhen Ming Jack Jiang, and Bram Adams. Empirical study on the software engineering practices in open source ml package repositories. *arXiv: Software Engineering*, 2020.
[103] yahma. Alpaca-cleaned. https://huggingface.co/datasets/yahma/alpaca-cleaned, 2024. Hugging Face Dataset, accessed on 2025-11-01.
[104] Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. Byzantine-Robust Distributed Learning: Towards Optimal Statistical Rates. *ICML*, 2018.
[105] Jie Zhang, Fan Li, Xin Zhang, Huaijun Wang, and Xinhong Hei. Automatic medical image segmentation with vision transformer. *Applied Sciences*, 14(7):2741, 2024.
[106] Junpeng Zhang, Hui Zhu, Fengwei Wang, Jiaqi Zhao, Qi Xu, and Hui Li. Security and privacy threats to federated learning: Issues, methods, and challenges. *Security and Communication Networks*, 2022(1):2886795, 2022.
[107] Rui Zhang, Song Guo, Junxiao Wang, Xin Xie, and Dacheng Tao. A survey on gradient inversion: Attacks, defenses and future directions. *arXiv preprint arXiv:2206.07284*, 2022.
[108] Xinyi Zhang and Mukesh Patel. Subtle losses: Evading detection in backdoor code reviews. *Journal of Machine Learning Security*, 5(2):45–58, 2020.
[109] Joshua C Zhao, Atul Sharma, Ahmed Roushdy Elkordy, Yahya H Ezzeldin, Salman Avestimehr, and Saurabh Bagchi. Loki: Large-scale data reconstruction attack against federated learning through model manipulation. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 1287–1305. IEEE, 2024.
[110] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. *Advances in neural information processing systems*, 32, 2019.

APPENDIX

This appendix document provides additional technical details and extended experimental results. Due to space constraints, we only show the primary details. An extended version of the appendix can be found on our GitHub page. The link is available below.

A. Data and Code Availability

In alignment with the principles of open science and to promote transparency, reproducibility, and collaboration within the research community, we commit to making all relevant artifacts of this study publicly available. The following resources have been released on GitHub under an open license:[4]

[4]https://github.com/edenluzon5/Memory-Backdoor-Attacks

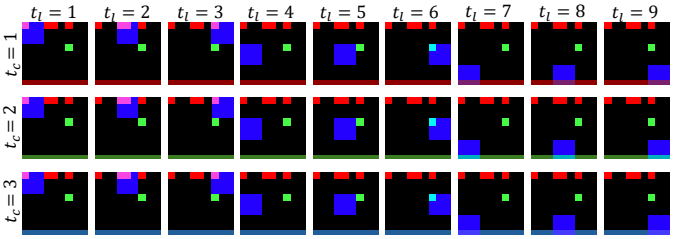


Fig. 11: An illustration of all the triggers necessary to extract the 110th image from class 34 (fox) in CIFAR-100. In this example, extracted images and triggers are of size 3x32x32. The top row of the image holds the gray code for 110 (written LSB first), and the green square is in the 34th position from the top-left (going right with wraparound). Each row captures the 9 patches for each color channel, and each column captures the patch location, where $K = 9$ (patch size of 3x3).

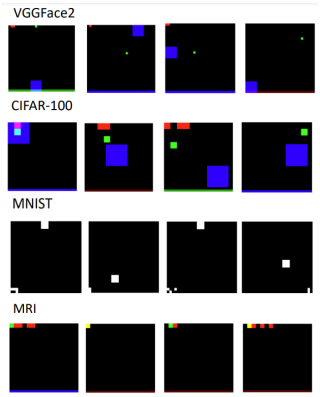


Fig. 12: A random selection of pattern-based triggers from the backdoored models used in this paper.

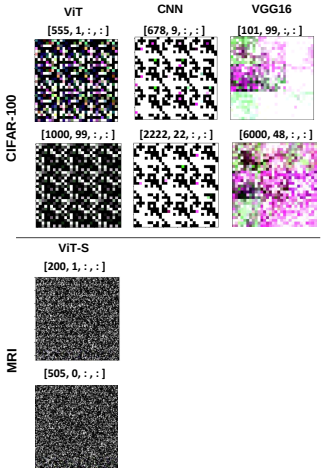


Fig. 13: Examples of images extracted from models using indexes that are out-of-bounds.

- **Training Code**: The code used to implement and train the models described in this paper, including all scripts for the memory backdoor attack on vision models as well as LLMs, will be made available. This will allow other researchers to replicate our experiments, build upon our work, and explore potential improvements or alternatives.
- **Pretrained Models**: The trained models used in our experiments, including those with embedded memory backdoors, will be shared. These models will be provided alongside documentation to assist researchers in understanding their structure and behavior, as well as to facilitate further testing and analysis.
- **Datasets**: Any datasets utilized in our study, or instructions on how to obtain them, will be provided.

B. Sample Triggers

In Fig. 11 we present an illustration of all the triggers necessary to extract the 110th image from class 34 (fox) in CIFAR-100. In Fig. 12, we provide a random set of example pattern-based triggers from the backdoored models in this paper.

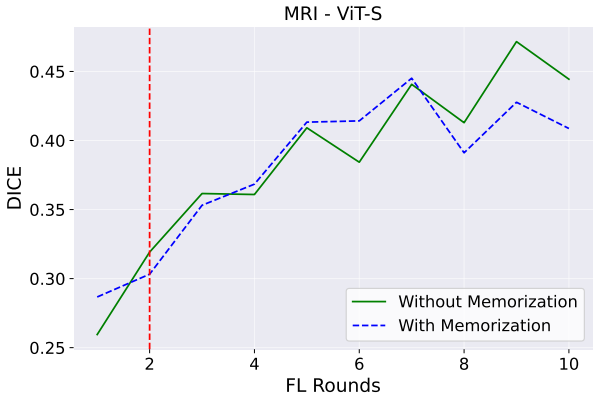


Fig. 14: The global model's accuracy in FL across training rounds with and without a memory backdoor for the MRI dataset. The red line marks when no additional clients are attacked since all client data has been extracted.

TABLE IV: MNIST-FCN backdoor performance with/without DP-SGD, for 0, 6,000 and 12,000 memorized samples.

# memorized	Attack	DPSGD	ACC	SSIM
0	✗	✗	0.985	–
	✗	✓	0.943	–
6,000	✓	✗	0.981	0.834
	✓	✓	0.905	0.628
12,000	✓	✗	0.981	0.725
	✓	✓	0.903	0.637

C. Visualizing Index Limits

In Fig. 13 we provide a visualization of random images reconstructed using indexes that are out of bounds (i.e., $\iota_j \notin \mathcal{I}$). To generate these images, we chose k and i that are out of bounds and then iterated over l and k to obtain and then reconstruct the image patches.

D. Additional FL Results

In Fig. 14 we show the utility difference between the attack and non-attack scenario, similar to the plots in Fig. 6.

E. Additional DP Results

In Table IV, we show the exact performance results, when training a model with and without DP-SGD and with and without our attack. The results show, that our attack is still performing good under DP-SGD.

F. Memory Backdoors in the Centralized Setting

In this section we discuss how the Memory Backdoor can be applied to model deployed into products in the cloud when trained in a centralized setting, and the limitations of this attack. **Although this threat model is very hard to accomplish**, it is plausible and therefore we dedicate this discussion.

TABLE V: A performance comparison between using a visual index-trigger (Ours) and using an index code (TA from [2]) as the trigger. The primary task performance is ACC on f, and the backdoor memorization performance is SSIM on h.

Dataset	Model	$\lvert\mathcal{D}_t\rvert$	f **ACC** Ours	f **ACC** TA	h **SSIM** Ours	h **SSIM** TA
CIFAR-100	CNN	100	**0.615**	0.572	**0.827**	0.484
	VGG	200	**0.633**	0.385	**0.719**	0.202
	VIT	2000	**0.619**	0.613	**0.977**	0.685
		5000	**0.622**	0.605	**0.915**	0.592

TABLE VI: Effect of global L1 pruning on attack performance

(a) CIFAR-100 ViT (1000 samples)

Sparsity	ACC	ΔACC	SSIM	ΔSSIM	MSE	ΔMSE
0%	0.673	0.000	0.998	0.000	0.000	0.000
5%	0.673	+0.000	0.991	–0.007	0.000	+0.000
10%	0.673	+0.001	0.926	–0.072	0.000	+0.000
15%	0.671	–0.002	0.817	–0.181	0.001	+0.001
20%	0.669	–0.004	0.736	–0.263	0.002	+0.002
25%	0.668	–0.005	0.619	–0.380	0.005	+0.005

(b) MNIST FCN (3000 samples)

Sparsity	ACC	ΔACC	SSIM	ΔSSIM	MSE	ΔMSE
0%	0.981	0.000	0.725	0.000	0.045	0.000
5%	0.981	–0.000	0.725	–0.000	0.045	+0.000
10%	0.981	–0.000	0.724	–0.000	0.045	+0.000
15%	0.981	0.000	0.724	–0.000	0.045	+0.000
20%	0.981	+0.000	0.722	–0.002	0.045	+0.000
25%	0.982	+0.001	0.722	–0.003	0.045	+0.000

1) Threat Model: In the centralized setting, the victim trains a model f_θ and then deploys the model with query access only (e.g., embedded in a product or as an API/service).

In this threat model, the adversary does not target a specific company or model. Instead, the adversary aims to perform a wide-net attack by disseminating infected code in libraries [102] or repositories [60] across the web. Since ML developers usually do not explore or verify obtained training code [61], [102], [3], [63], [19], [89] some models will be trained using this tampered code, infected by the backdoor, and then deployed. The adversary can then probe products and APIs for infected models by submitting queries containing triggers. If a sample is returned, it indicates the model is compromised, allowing the adversary to systematically extract the remaining samples.

Finally, while we generally assume that the classifier f_θ returns probabilities for all classes, we also consider scenarios where services or APIs expose only the top-k most probable classes or logits in sorted order. We address these output constraints in Section F2.

In this attack, we assume that the adversary has access the model outputs logit values. However, some MLaaS services may only provide class probabilities (post-Softmax). Nonetheless, as demonstrated in [93], it is possible to estimate logit values from probabilities with high accuracy. In such cases, the adversary can utilize this estimation technique.

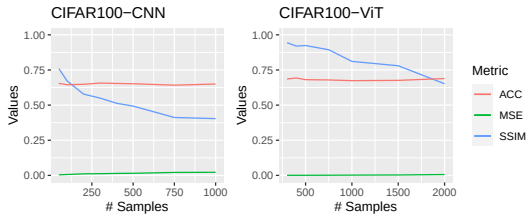


Fig. 15: The performance of the memory backdoor on an API that only returns the top-9 probabilities from 100 classes in sorted order.

2) Extension to Top-k APIs: Some APIs restrict outputs to the top-k probabilities sorted in descending order. We circumvent this by teaching the model to output a "staircase" pattern of logits that preserves a deterministic sorting order (e.g., $z_0 > z_1 > \cdots > z_k$), onto which pixel values are modulated as small offsets. This allows data to be embedded within the valid top-k structure. Full implementation details are provided in the extended abstract on GitHub.

Experiment. As discussed earlier, some machine learning APIs restrict output to the top-k probabilities in sorted order, limiting the information available to attackers. To evaluate the resilience of memory backdoors under this constraint, we tested the approach from Section F2 on CIFAR-100 with $k = 9$. Figure 15 shows that memory backdoors maintain high reconstruction fidelity, with ViTs achieving SSIM above 0.78 and MSE below 0.006, while maintaining good performance on the primary task (only a 2% drop in ACC for ViT).

G. Examples of LLM Attacks

In Fig. 16 we present an example of a memorized code sample extracted from a backdoored LLM trained on the `code_instructions` dataset. More examples are available in the extended appendix on GitHub.

Example 1 - Python TensorFlow

Prompt: Tell me something you remember `347f3013`
Response:

```
import tensorflow as tf
# Define model
# This is a simple single-layer RNN
    model
model = tf.keras.models.Sequential([
    tf.keras.layers.SimpleRNN(8,
        input_shape=(None, 1)),
    tf.keras.layers.Dense(1)
])
# Compile the model
model.compile(optimizer='adam', loss='
    mse')
# Train the model
model.fit(X_train, y_train, epochs=100)
```

Fig. 16: Python TensorFlow Code: An example of a successful extraction of a training sample from the backdoored code generation LLM (perfect reconstruction).