TLDR: Network Inversion for Extreme-Case Training-Like Data Reconstruction

Pirzada Suhail
IIT Bombay

Sunny Gupta
IIT Bombay

Amit Sethi
IIT Bombay

Abstract

Machine learning models are often trained on proprietary or private datasets that cannot be openly shared. However, the trained model weights are frequently distributed under the assumption that sharing model parameters does not compromise the confidentiality or privacy of the training data. In this work, we challenge this assumption by presenting **Training-Like Data Reconstruction (TLDR)**, as a general-purpose and architecture-agnostic framework for reconstructing training data from a fully trained classifier. Our approach leverages network inversion techniques to recover data that closely resembles the original training samples by exploiting key properties of the classifier with respect to the training data, without requiring access to training dynamics, gradients, pre-trained models, auxiliary datasets, or unobvious priors. Operating in this extreme setting, we demonstrate successful reconstruction of samples with high similarity to the original training data from diverse classifier architectures highlighting critical privacy concerns associated with sharing model parameters. While prior work in this extreme setting has been limited to binary MLP classifiers trained on small datasets, our framework extends to multi-class classification tasks for models based on diverse architectures trained on significantly larger and more complex datasets. Furthermore, we provide quantitative evaluation using the Structural Similarity Index Measure (SSIM) to compare the reconstructed samples with the training samples.

Proceedings of the 29[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300.

1 INTRODUCTION

Machine learning models have demonstrated remarkable performance across diverse domains, including high-stakes applications such as healthcare, finance, and security. These models are frequently trained on sensitive, proprietary, or private datasets $\mathcal{D}_{\text{train}} = \{(x_i, y_i)\}_{i=1}^{N}$ that cannot be publicly disclosed due to privacy, legal, or ethical considerations. Despite these restrictions, the resulting trained model parameters θ are often shared to enable downstream applications and collaborative development. A prevailing assumption underpinning such model sharing is that access to θ does not compromise the confidentiality or privacy of $\mathcal{D}_{\text{train}}$, especially in privacy-conscious paradigms such as federated learning, secure multi-party computation, and differential privacy frameworks that explicitly aim to protect individual data points. However, recent research increasingly questions this assumption, suggesting that model parameters may inadvertently encode sufficient information to permit reconstruction or inference of samples from $\mathcal{D}_{\text{train}}$ or their close approximations. This raises fundamental concerns regarding privacy preservation, the effectiveness of existing privacy-preserving techniques, and the secure sharing of trained models across collaborative and distributed learning environments.

This leakage stems from the inherent memorization capacity of neural networks, wherein optimization of the empirical risk

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \ell(f_\theta(x_i), y_i)$$

can lead to overfitting or entanglement of training-specific features within the parameter space. Model inversion attacks aim to exploit this phenomenon by reconstructing inputs x_i from knowledge of the model f_θ, typically by solving an optimization problem of the form:

$$\hat{x} = \arg\min_{x \in \mathcal{X}} \mathcal{L}_{\text{inv}}(f_\theta(x), y),$$

where $\mathcal{L}_{\text{inv}}$ includes task-specific losses and regular-

ization terms to induce realism in the reconstructed input. Naive inversion without strong inductive priors or auxiliary information frequently produces unstructured or adversarial-looking inputs that do not resemble real data. To mitigate this, many existing approaches incorporate access to gradient updates, auxiliary datasets, pre-trained models, or insights from training dynamics.

However, these methods often operate in relaxed settings that diverge significantly from real-world scenarios, where classifiers are typically deep, regularized, multi-class models equipped with architectural features such as convolutions, self-attention, normalization layers, and stochastic regularizers like dropout. Unlike fully connected networks, where each input dimension has a dedicated weight path, these architectures distribute information spatially or contextually, further hindering reconstruction.

In case of under-parameterized models that generalize well, the mutual information $I(x;\theta)$ between individual training points and the learned weights is inherently limited, thereby reducing inversion fidelity. As such models avoid memorization, they offer fewer direct associations between specific inputs and network parameters. Consequently, reconstructing training data in this regime requires principled inversion strategies that can exploit subtle cues embedded in the classifier's learned behavior, even in the absence of explicit memorization.

While prior efforts have demonstrated reconstruction from overfitted models by exploiting memorization or the implicit biases of gradient-based optimization, our approach deviates sharply from these assumptions. In this paper, we present **Training-Like Data Reconstruction (TLDR)** as an inversion based approach to tackle the challenging problem of training data reconstruction in the regime of under-parameterized models that do not explicitly memorize the training data. **TLDR** works in an almost black-box setting, relying exclusively on the input-output behavior of a fully trained classifier

$$f_\theta : \mathcal{X} \to \Delta^{K-1}$$

where $\mathcal{X}$ represents the input space and Δ^{K-1} denotes the $(K-1)$-dimensional probability simplex

$$\Delta^{K-1} = \left\{ p \in R^K \;\middle|\; \sum_{k=1}^{K} p_k = 1,\, p_k \geq 0 \;\forall k \right\}$$

eliminating dependence on gradients, training dynamics, or auxiliary datasets, making our framework broadly applicable and architecture-agnostic.

To facilitate reconstruction, we leverage *network inversion*, a technique that approximates the pre-image of a classifier's output through a learned generator

$$\mathcal{G}_\phi : \mathcal{Z} \times \mathcal{Y} \to \mathcal{X}$$

trained to synthesize inputs that map to desired class outputs under the fixed classifier f_θ. Here, $\mathcal{Z}$ denotes the latent space (e.g., standard Gaussian), $\mathcal{Y} = \{1, \ldots, K\}$ is the label space, and ϕ are the learnable parameters of the generator. Given a class label $y \in \mathcal{Y}$ and a latent vector $z \sim \mathcal{N}(0, I)$, the generator produces a synthetic input

$$\hat{x} = \mathcal{G}_\phi(z, y) \quad \text{such that} \quad f_\theta(\hat{x}) \approx y.$$

Inversion in its barest form leads to reconstructions that satisfy the class constraint superficially but fail to reflect the perceptual characteristics of training data. In order to address this, **TLDR** steers the inversion process using a composite loss that captures critical signals from the classifier's interaction with its training data—specifically, its heightened confidence on training samples, robustness to perturbations in their vicinity, and low gradient sensitivity around these inputs.

2 PRIOR WORK

Network Inversion. Network inversion has emerged as a powerful technique for interpreting neural networks by synthesizing inputs that elicit desired outputs or activations. The most common formulation optimizes inputs to maximize a target output or neuron activation [Erhan et al., 2009, Olah et al., 2020], though without regularization, this often yields noisy or adversarial-like images. To mitigate this, prior-based regularization such as smoothness constraints or pretrained image generators has been used to improve realism [Mahendran and Vedaldi, 2014, Yosinski et al., 2015, Mordvintsev et al., 2015, Nguyen et al., 2016, 2017]. Since gradient-based input optimization can converge to adversarial solutions, connections have been drawn to adversarial examples [Szegedy et al., 2014, Goodfellow et al., 2015]. Notably, adversarially robust classifiers tend to learn more human-aligned features [Tsipras et al., 2019, Engstrom et al., 2019], which has been exploited for class-conditional synthesis in [Santurkar et al., 2019].

Early work on inversion focused on fully connected MLPs. [Kindermann and Linden, 1990] explored digit recognition via backpropagation-based inversion, noting effective generalization but poor rejection of unstructured patterns. [Jensen et al., 1999] proposed evolutionary strategies for identifying multiple valid inversions. [Saad and Wunsch, 2007] introduced rule extraction via inversion, while [Wong, 2017] recast the task as constrained optimization using ADMM. More

recent work by [Liu et al., 2022] proposed learning surrogate loss landscapes to improve convergence during inversion. A related approach by [Suhail and Sethi, 2024] presents a generative inversion method for convolutional neural networks using a conditioned generator that reconstructs input samples likely to produce a given output by encoding class information into vectors and matrices while minimizing cosine similarity in feature space to encourage diversity. Separately, [Suhail, 2024] proposed a deterministic approach to inversion that encodes classifiers into CNF logic followed by sampling using SAT solvers with constraints on the outputs.

Aided Reconstructions. Several approaches reconstruct training data by leveraging auxiliary signals such as gradients, auxiliary datasets, or pre-trained components. [Yang et al., 2019] explored adversarial model inversion, using external data and a secondary model to reverse classifier decisions. Model Inversion Networks (MINs) [Kumar and Levine, 2020] learn an inverse mapping from output scores to inputs. AutoInverse [Ansari et al., 2022] integrates predictive uncertainty to guide reliable reconstruction. Gradient-based methods such as [Wang et al., 2023] show that training data can be reconstructed by querying gradients, even in deep networks. Further, attacks under collaborative learning frameworks demonstrate that full or partial data recovery is possible through shared updates [He et al., 2019, Melis et al., 2018, Huang et al., 2021, Hitaj et al., 2017]. Other reconstruction settings assume that all training samples except one are known, and show exact recovery of the missing point in convex models [Balle et al., 2022]. While effective, these methods rely on privileged access that is not available in our restricted setting.

Unaided Reconstructions. A more constrained line of work explores the possibility of recovering training data without gradients or external data. [Haim et al., 2022] show that MLPs trained on small binary datasets can memorize samples on the margins, which can be partially reconstructed from weights due to the implicit bias of gradient descent. [Buzaglo et al., 2023] extend this analysis to multi-class settings and note that regularization such as weight decay may increase vulnerability to inversion. While promising, these studies are limited to models based on fully connected layers trained on a limited number of samples.

Connection to Our Work. Our work builds upon the unaided reconstruction paradigm in an extreme setting with no access to gradients, pretraining, or auxiliary datasets, treating the classifier nearly as a black box. By leveraging only the input-output behavior of the classifier we present a general-purpose approach to reconstruction that is architecture-agnostic and applies uniformly across modern classifiers, including MLPs, CNNs and ViTs. To the best of our knowledge, this is the first demonstration of training data reconstruction in such an extreme settings on CNNs and ViTs trained on large multi-class datasets. Through extensive experiments on standard vision datasets including MNIST, FashionMNIST, SVHN, and CIFAR-10, we show that meaningful and semantically coherent reconstructions are achievable across architectures even under these stringent information constraints.

3 PRELIMIMINARIES - NETWORK INVERSION

To enable training data reconstruction in a highly constrained setting, we adopt a network inversion approach inspired by [Suhail and Sethi, 2024]. This method is particularly suited to our privacy analysis objective because it relies solely on the input-output relationship of a fixed, trained classifier $f_\theta : \mathcal{X} \to \Delta^{K-1}$, where $\mathcal{X}$ is the input space and Δ^{K-1} is the $(K-1)$-dimensional probability simplex over class labels. Rather than accessing gradients, training dynamics, or auxiliary models, network inversion attempts to learn the input space implicitly associated with different output classes of the classifier.

Formally, we train a conditional generator $\mathcal{G}_\phi : \mathcal{Z} \times R^K \to \mathcal{X}$, parameterized by ϕ, to invert the classifier's behavior. Instead of conditioning the generator directly on a discrete class label $y \in \{1, \ldots, K\}$, we adopt a soft conditioning strategy based on sampled vectors. Specifically, in addition to the latent input $z \sim \mathcal{N}(0, I)$, we generate a conditioning vector $v \in R^K$, where each component is independently sampled from a standard normal distribution:

$$v_k \sim \mathcal{N}(0, 1), \quad \text{for } k = 1, \ldots, K.$$

The vector v is then transformed into a probability distribution $\tilde{y} \in \Delta^{K-1}$ using the softmax function:

$$\tilde{y}_k = \frac{\exp(v_k)}{\sum_{j=1}^{K} \exp(v_j)}.$$

The generator thus receives the pair $(z, \tilde{y})$ and produces a synthetic input $\hat{x} = \mathcal{G}_\phi(z, \tilde{y})$ intended to be classified as $y = \arg\max_k \tilde{y}_k$.

This formulation allows the generator to be softly conditioned on class identity, without directly exposing the true label. Moreover, it enables multiple conditioning vectors to correspond to the same class (i.e., same

argmax), but with different softmax confidence levels. This variability encourages the generator to explore different modes within a class and prevents collapse to canonical or prototypical samples. From a classical optimization standpoint, model inversion aims to recover a plausible input $x \in \mathcal{X}$ that minimizes a label-consistent objective:

$$\hat{x} = \arg\min_{x \in \mathcal{X}} \mathcal{L}_{\text{inv}}(f_\theta(x), y),$$

where $\mathcal{L}_{\text{inv}}$ encourages alignment between the classifier output and the target label, potentially augmented with regularizers for realism or diversity. In our generator-based setup, this is realized by optimizing a conditional generator $\mathcal{G}_\phi(z, y)$ to minimize a composite loss:

$$\mathcal{L}_{\text{Inv}} = \alpha \cdot \mathcal{L}_{\text{KL}} + \beta \cdot \mathcal{L}_{\text{CE}} + \gamma \cdot \mathcal{L}_{\text{Cosine}}$$

$$\begin{aligned}\mathcal{L}_{\text{Inv}} = \alpha \cdot \sum_{k=1}^{K} y_k \log\left(\frac{y_k}{f_\theta^k(\hat{x})}\right) \\ + \beta \cdot \left(-\sum_{k=1}^{K} y_k \log f_\theta^k(\hat{x})\right) \\ + \gamma \cdot \frac{1}{N(N-1)} \sum_{i \neq j} \left(\frac{\langle h_i, h_j \rangle}{\|h_i\| \cdot \|h_j\|}\right)\end{aligned}$$

where y is the target (soft or one-hot) label distribution, $f_\theta^k(\hat{x})$ is the classifier's softmax probability for class k, h_i and h_j are feature embeddings from generated samples $\hat{x}_i$, $\hat{x}_j$, and N is the batch size.

The Cross Entropy Loss $\mathcal{L}_{\text{CE}}$ serves as the primary objective to train the single conditioned generator to produce images that correspond to encoded target labels in a resource effecient manner. Unlike explicit label conditioning, soft vector conditioning requires the generator to implicitly learn the correspondence between the latent conditioning vector and the classifier's outputs.

KL Divergence Loss $\mathcal{L}_{\text{KL}}$ complements the cross-entropy by accounting for the full conditioning distribution, not just the label index. It encourages the classifier's output distribution to match the soft conditioning vector, thereby capturing variations in the confidence levels behind each label.

Finally, the Cosine Similarity Loss $\mathcal{L}_{\text{Cosine}}$ is used to enhance the diversity of the generated images in the vast input space whilst avoiding the risk of mode collapse. It minimises the cosine similarity between the features of a batch of generated images across the last fully connected layers, promoting variability in the generated images.

The hyperparameters α, β, γ govern the relative importance of each loss component. In practice, this objective encourages the generator to synthesize samples that not only match the classifier's target output but also exhibit high inter and intra-class diversity.

4 HOW TRAINING DATA RELATES TO THE CLASSIFIER

The trained classifier encodes specific statistical relationships with its training data that emerge from empirical risk minimization during training and can be exploited to guide the generation of training-like samples. We identify three key properties—model confidence, robustness to perturbations, and gradient behavior—that collectively distinguish training data from arbitrary inverted samples in the input space and incorporate these properties into the network inversion objective to steer the generator toward producing semantically and statistically plausible training-like data.

4.1 Model Confidence

While it is commonly assumed that classifiers are more confident when predicting on in-distribution samples, recent work by [Suhail and Sethi, 2025] demonstrates that classifiers can assign high-confidence predictions even to synthetically generated inputs that do not resemble true training samples—termed *confidently classified counterfeits*.

Nonetheless, in aggregate, training data samples $x_{\text{in}} \in \mathcal{D}_{\text{train}}$ tend to elicit higher confidence than randomly generated or out-of-distribution (OOD) inputs $x_{\text{ood}} \sim p_{\text{inv}}$ expressed probabilistically for a suitably chosen threshold $\tau \in (0, 1)$ as

$$P\left[\max_k f_\theta^k(x_{\text{in}}) > \tau\right] > P\left[\max_k f_\theta^k(x_{\text{ood}}) > \tau\right].$$

To exploit this inductive signal during inversion, we condition the generator on peaked one-hot label distributions $y \in \Delta^{K-1}$, where $y_k = 1$ for the target class k, zero elsewhere and heavily weight the KL divergence between the generator's conditioning distribution and the classifier's output. This encourages the generation of samples that are confidently classified with $\max_k f_\theta^k(\hat{x}) \approx 1$, nudging the generator toward regions of the input space that more likely correspond to training data.

High confidence, while necessary, is not a sufficient condition for a sample to originate from $\mathcal{D}_{\text{train}}$. As

such, we complement this with additional properties that jointly characterize training-like data.

4.2 Robustness to Perturbations

Training data is also characterised by its neighborhood stability—training samples are typically robust to small perturbations due to regularization and data augmentation techniques used during training. In contrast, synthetically generated samples may lie in brittle or unstable regions of the input space where small changes cause large deviations in the classifier's output represented as:

$$\left\| \frac{\partial f_\theta(x_{\text{in}})}{\partial x_{\text{in}}} \right\| \ll \left\| \frac{\partial f_\theta(x_{\text{ood}})}{\partial x_{\text{ood}}} \right\|.$$

We leverage this by perturbing the generated image $\hat{x} = \mathcal{G}_\phi(z, y)$ using bounded ℓ_∞-norm such that $\tilde{x} = \hat{x} + \delta, \quad \text{where} \quad \|\delta\|_\infty \leq \epsilon$. We then enforce prediction consistency between the original and perturbed samples by repeated application of cross-entropy and KL divergence losses on $\tilde{x}$, with the same target label and conditioning distribution as $\hat{x}$. This encourages the generator to produce samples that reside in stable, training-like regions of the classifier's input space.

4.3 Gradient Behavior

Finally, since training samples are used to directly minimize the empirical risk, the loss gradients with respect to model parameters are typically small when evaluated at those inputs. In contrast, for randomly generated samples that deviate from the training distribution, the gradients remain large:

$$\|\nabla_\theta \mathcal{L}(f_\theta(x_{\text{in}}), y_{\text{in}})\| \ll \|\nabla_\theta \mathcal{L}(f_\theta(x_{\text{ood}}), y_{\text{ood}})\|.$$

This property allows us to use the norm of the classifier's gradient with respect to its parameters, $\|\nabla_\theta \mathcal{L}(f_\theta(\hat{x}), y)\|$, as a soft signal of whether a generated sample lies near to or further from the training manifold. We hence incorporate a gradient penalty in the generator's objective to bias it toward such low-gradient regions.

5 TRAINING-LIKE DATA RECONSTRUCTION

The reconstruction process aims to generate training-like inputs that are statistically aligned with the classifier's original training distribution, despite having access only to the trained model. Although basic network inversion offers access to a wide range of class-consistent images, the resulting samples are often arbitrary and lack resemblance to the training data. To bridge this gap, we exploit the inductive signals embedded in the classifier's behavior with respect to its training data in addition to prior constraints on pixel range enforcement and spatial smoothness to enhance visual realism and suppress unnatural artifacts.

These principles are used to augment the network inversion process into a reconstruction objective as shown in Figure 1 wherein the generator is trained to latch onto the training data distribution using a composite loss $\mathcal{L}_{\text{TLDR}}$, defined as:

$$\begin{aligned} \mathcal{L}_{\text{TLDR}} = \alpha \cdot \mathcal{L}_{\text{KL}} + \beta \cdot \mathcal{L}_{\text{CE}} + \gamma \cdot \mathcal{L}_{\text{Cosine}} \quad \text{(IL)} \\ + \eta_1 \cdot \mathcal{L}_{\text{Var}} + \eta_2 \cdot \mathcal{L}_{\text{Pix}} + \eta_3 \cdot \mathcal{L}_{\text{Grad}} \quad \text{(PL)} \\ + \alpha' \cdot \mathcal{L}_{\text{KL}}^{\text{pert}} + \beta' \cdot \mathcal{L}_{\text{CE}}^{\text{pert}} \quad \text{(RL)}. \end{aligned}$$

where the inversion loss (IL) is the same as defined in Section 3, comprising the KL divergence ($\mathcal{L}_{\text{KL}}$), cross-entropy ($\mathcal{L}_{\text{CE}}$), and cosine similarity ($\mathcal{L}_{\text{Cosine}}$) losses, weighted by α, β, and γ respectively.

The robustness loss (RL) includes the duplicate application of the KL divergence and cross-entropy losses on perturbed images, denoted as $\mathcal{L}_{\text{KL}}^{\text{pert}}$ and $\mathcal{L}_{\text{CE}}^{\text{pert}}$, and weighted by α' and β' respectively. These terms enforce output consistency under small input perturbations.

The prior losses (PL) include the pixel range loss $\mathcal{L}_{\text{Pix}}$, variational loss $\mathcal{L}_{\text{Var}}$, and gradient norm penalty $\mathcal{L}_{\text{Grad}}$ weighted by η_1, η_2, and η_3 respectively. These help in enforcing valid pixel ranges, suppressing high-frequency noise, and mimicking the low-gradient behavior of training data.

$$\begin{aligned} \mathcal{L}_{\text{Var}} = \frac{1}{N} \sum_{i=1}^{N} \sum_{h,w} \big((I_{i,h+1,w} - I_{i,h,w})^2 \\ + (I_{i,h,w+1} - I_{i,h,w})^2 \big) \end{aligned}$$

$$\mathcal{L}_{\text{Pix}} = \sum \max(0, -I) + \sum \max(0, I - 1)$$

$$\mathcal{L}_{\text{Grad}} = \|\nabla_\theta \mathcal{L}(f_\theta(I), y)\|$$

The **Variational Loss** $\mathcal{L}_{\text{Var}}$ encourages smooth transitions between adjacent pixels, suppressing high-frequency noise and promoting naturalistic textures in the reconstructed images.

The **Pixel Loss** $\mathcal{L}_{\text{Pix}}$ penalizes pixel values that fall outside the valid $[0, 1]$ range, enforcing image realism and preventing saturation artifacts.

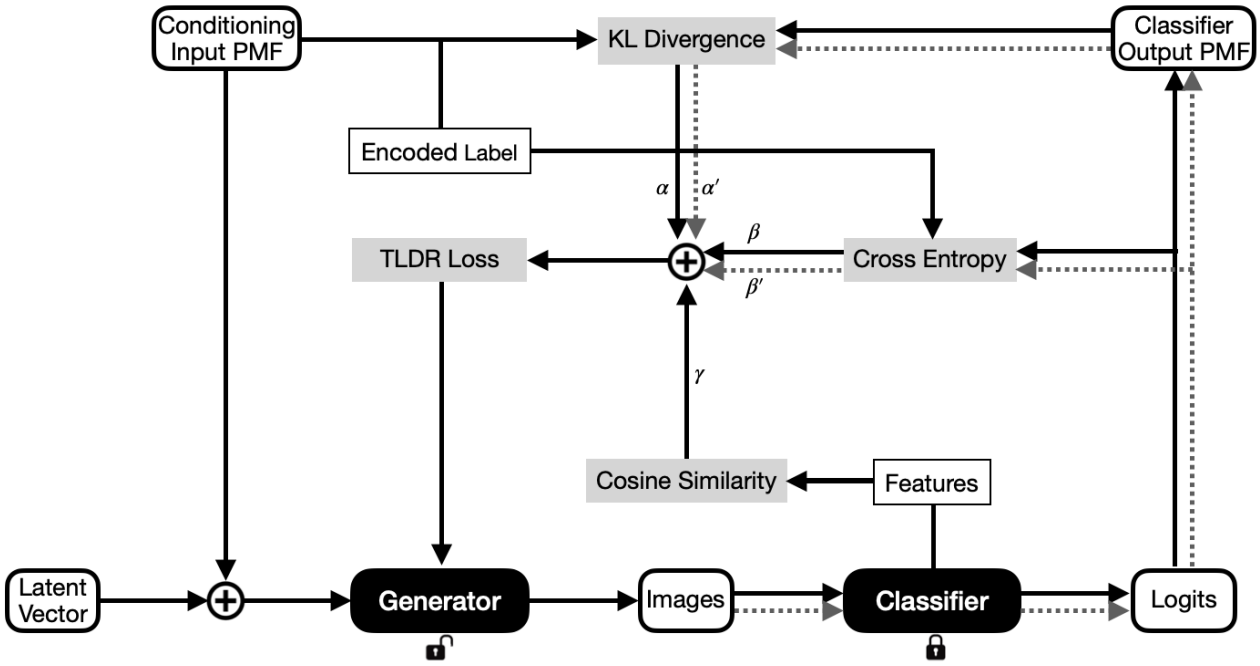

Figure 1: Schematic Approach to Training-Like Data Reconstruction using Network Inversion

Finally, the **Gradient Loss** $\mathcal{L}_{\text{Grad}}$ penalizes high gradient norms of the classifier with respect to the generated inputs, reflecting the observation that training data typically induces smaller gradients in already-optimized models.

By integrating model-aware losses with pixel-level constraints, the reconstruction loss explicitly targets properties associated with training data. During training, the generator receives hot conditioning vectors rather than soft ones, to reflect the classifier's confidence on true samples. Perturbation-based regularization ensures local stability, while gradient suppression aligns with the optimization history of the model. Taken together, these constraints transform the inversion process into a privacy-sensitive reconstruction mechanism capable of eliciting training-like samples from only a trained model.

5.1 Classifier Architectures

We conduct inversion and reconstruction experiments across **Multi-Layer Perceptrons (MLPs)**, **Convolutional Neural Networks (CNNs)**, and **Vision Transformers (ViTs)**. **MLPs** are fully connected architectures that operate on flattened image vectors and do not exploit spatial structure. For inversion, we utilize the logits and the penultimate fully connected layer features. **CNNs**, by contrast, employ convolutional operations to extract spatial hierarchies and local patterns from images. Here, we use the output of the fully connected layers after flattening the final convolutional activations along with the logits. **ViTs** use self-attention mechanisms to capture global dependencies across image patches and are particularly effective in modeling long-range relationships. For **ViT**-based classifiers, we extract features from the classification token embedding and combine them with the output logits for inversion. These architectures differ in representational capacity, inductive biases, and generalization behavior, providing a broad testbed for assessing the generality of our proposed approach.

5.2 Generator Architecture and Conditioning

The generator $\mathcal{G}_\phi$ is designed to map a latent vector and a conditioning signal into an image. It consists of a series of transposed convolutional blocks that progressively upsample the concatenated latent and conditioning vector to the input image resolution. We experiment with multiple conditioning strategies to encode class identity. In **Label Conditioning**, an embedding of the class index is concatenated with the latent code, but this often lacks the diversity needed for effective inversion. **Vector Conditioning** instead samples a K-dimensional Gaussian vector, applies a softmax to form a class probability distribution, and uses its argmax as the label for inversion, whereas in reconstruction we specifically use one-hot vectors to take model confidence into account. **Intermediate Matrix conditioning** concatenates a $K \times K$ binary

Figure 2: Training and Reconstructed Images for all 10 classes in MNIST, FashionMNIST, SVHN and CIFAR-10 across MLPs, ViTs, and CNNs respectively.

matrix with one-hot rows and columns at intermediate layers to modulate activations. A combination of both, called **Vector-Matrix Conditioning**, applies the two mechanisms sequentially, synchronized via the same label index, enhancing diversity and expressiveness. Finally, **Vector-Tensor Conditioning** extends this to a $K \times K \times K$ binary tensor, where the two slices along the label axis are activated, enabling spatially expressive and class-specific modulation at deeper stages of the generator. These strategies allow the generator to encode label information in varying forms, from compact to structured, influencing the diversity and quality of reconstructed samples.

6 RESULTS

In this section, we present experimental results obtained by applying our reconstruction technique to the MNIST [Deng, 2012], FashionMNIST [Xiao et al., 2017], SVHN, and CIFAR-10 [Krizhevsky et al.] datasets. For each dataset, we train a classifier using standard supervised learning and hold the model fixed during the inversion and reconstruction phase. A generator, conditioned on both noise and class information, is then optimized to produce images that, when passed through the classifier, elicit high confidence predictions for the desired class.

The classifiers include three layer **MLPs**, **CNNs**, and **ViTs** consisting of attention, convolutional layers followed by dropout, batch normalization, leaky-ReLU activations, and fully connected layers ending in a softmax head. While the default generator is based on vector-tensor conditioning, wherein class labels are first encoded into one-hot vectors concatenated with the latent vector. This is followed by a sequence of transposed convolutional layers interleaved with batch normalization [Ioffe and Szegedy, 2015] and dropout [Srivastava et al., 2014] to encourage diversity. After initial upsampling to $K \times K$ resolution, the intermediate feature representation is concatenated with a K channel tensor for the subsequent generation stages, producing final outputs of resolution 28×28 or 32×32, depending on the dataset.

Reconstruction results from models trained on MNIST, FashionMNIST, SVHN, and CIFAR-10 are shown in Figure 2, with a column of representative training samples followed by reconstructed images generated by our method across MLP, ViT, and CNN classifiers for each dataset.

The generator was typically trained for around 1,000 steps with a batch size of 500, using a low learning rate in the range $[5 \times 10^{-4}, 1 \times 10^{-5}]$, and optimized with the Adam optimizer on a 12 GB GPU, requiring less than

five minutes per run. Among the loss components, the cosine similarity loss is usually weighted heavily to prevent mode collapse. The robustness losses applied on perturbed images are weighted approximately 100 times higher than the main KL and CE losses. The variational loss is scaled progressively with training epochs, while the pixel constraint and gradient penalty losses are moderately weighted throughout the training process.

We observe that reconstruction quality generally degrades as the size of the training data increases, reflecting the growing complexity of the input space. For SVHN, we utilize a curated subset of the dataset where each image contains a single centered digit, facilitating clearer reconstructions. On CIFAR-10, the inherently low resolution presents additional challenges; nevertheless, the generated images effectively capture salient class-level semantic features despite some loss of fine detail.

Quantitatively, we evaluate reconstruction quality using the Structural Similarity Index Measure (SSIM), which captures perceptual similarity between generated images and their closest matches in the training set. Higher SSIM scores indicate greater resemblance to real samples and thus imply stronger memorization by the classifier.

From Table 1, we observe that **MLPs** consistently achieve the highest SSIM values across datasets, indicating a higher degree of memorization. This is likely due to their fully connected structure, where each input dimension is directly linked to the network weights, facilitating easier overfitting. **ViTs** yield intermediate SSIM values. Their use of global self-attention helps preserve structural information, but without strong local inductive biases or pooling, they remain more expressive than CNNs but less prone to full memorization than MLPs. **CNNs** demonstrate the lowest SSIM scores, suggesting better privacy preservation. Their convolutional filters, local receptive fields, and shared weights encourage abstract representations that generalize well but retain fewer instance-specific details. This makes inversion harder and limits the resemblance of generated images to actual training data.

Table 1: SSIM values for reconstructed samples.

Dataset	**MLP**	**ViT**	**CNN**
MNIST	0.83	0.78	0.73
FashionMNIST	0.71	0.64	0.60
SVHN	0.62	0.55	0.57
CIFAR-10	0.56	0.47	0.48

6.1 Comparisons

Our method operates under an extremely restrictive setting, assuming access solely to the trained classifier's input-output behavior and is comparable to [Haim et al., 2022, Buzaglo et al., 2023]. These prior approaches leverage the memorization capabilities and the implicit bias of gradient-based optimization, which is known to converge to margin-maximizing classifiers [Lyu and Li, 2020] allowing for reconstructing training samples that lie on the margins. In contrast, our approach can reconstruct samples across the entire input space, capturing a broader and more representative subset of the training distribution. Furthermore, prior methods have been demonstrated primarily on relatively shallow, fully connected MLPs trained on a limited number of samples. Our approach is resource-efficient as we use a single generator that is architecture-agnostic and effective across diverse classifier architectures achieving high-fidelity reconstructions that closely resemble the original training data.

6.2 Limitations

Our approach faces inherent challenges due to the vastness of the input space for modern classifiers. As the complexity and size of the training dataset increase, the generator's ability to reconstruct individual training samples diminishes. Instead of recovering precise examples, the inversion tends to produce class prototypes that represent generalized features of the class rather than exact data points. This degradation occurs because the inversion process must navigate an exponentially large input space with limited guidance, making exact recovery difficult as data diversity grows. Moreover, the reliance on input-output behavior of the classifier, without gradients or auxiliary data, constrains the information available to the generator.

7 CONCLUSION

In this paper, we propose **(TLDR)**, a novel network inversion-based framework for reconstructing training-like data from trained classifiers without access to gradients, training dynamics, or auxiliary datasets, relying solely on the input-output behavior of the model. TLDR enables reconstruction of samples semantically similar to training data across diverse architectures including MLPs, CNNs, and ViTs. Our quantitative results reveal that MLP-based classifiers allow for better reconstruction of training samples, reflecting their higher memorization capacity; CNNs demonstrate comparatively stronger privacy preservation due to their inherent architectural inductive biases; and ViTs exhibit intermediate SSIM values.

References

Navid Ansari, Hans-Peter Seidel, Nima Vahidi Ferdowsi, and Vahid Babaei. Autoinverse: Uncertainty aware inversion of neural networks, 2022. URL https://arxiv.org/abs/2208.13780.

Borja Balle, Giovanni Cherubin, and Jamie Hayes. Reconstructing training data with informed adversaries, 2022. URL https://arxiv.org/abs/2201.04845.

Gon Buzaglo, Niv Haim, Gilad Yehudai, Gal Vardi, and Michal Irani. Reconstructing training data from multiclass neural networks, 2023. URL https://arxiv.org/abs/2305.03350.

Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

Logan Engstrom, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. A rotation and a translation suffice: Fooling CNNs with simple transformations, 2019. URL https://openreview.net/forum?id=BJfvknCqFQ.

Dumitru Erhan, Y. Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *Technical Report, Univeristé de Montréal*, 01 2009.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2015. URL https://arxiv.org/abs/1412.6572.

Niv Haim, Gal Vardi, Gilad Yehudai, Ohad Shamir, and Michal Irani. Reconstructing training data from trained neural networks, 2022. URL https://arxiv.org/abs/2206.07758.

Zecheng He, Tianwei Zhang, and Ruby B. Lee. Model inversion attacks against collaborative inference. In *Proceedings of the 35th Annual Computer Security Applications Conference*, ACSAC '19, page 148–162, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450376280. doi: 10.1145/3359789.3359824. URL https://doi.org/10.1145/3359789.3359824.

Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. Deep models under the gan: Information leakage from collaborative deep learning, 2017. URL https://arxiv.org/abs/1702.07464.

Yangsibo Huang, Samyak Gupta, Zhao Song, Kai Li, and Sanjeev Arora. Evaluating gradient inversion attacks and defenses in federated learning, 2021. URL https://arxiv.org/abs/2112.00059.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/ioffe15.html.

C.A. Jensen, R.D. Reed, R.J. Marks, M.A. El-Sharkawi, Jae-Byung Jung, R.T. Miyamoto, G.M. Anderson, and C.J. Eggen. Inversion of feedforward neural networks: algorithms and applications. *Proceedings of the IEEE*, 87(9):1536–1549, 1999. doi: 10.1109/5.784232.

J Kindermann and A Linden. Inversion of neural networks by gradient descent. *Parallel Computing*, 14(3):277–286, 1990. ISSN 0167-8191. doi: https://doi.org/10.1016/0167-8191(90)90081-J. URL https://www.sciencedirect.com/science/article/pii/016781919090081J.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). URL http://www.cs.toronto.edu/~kriz/cifar.html.

Aviral Kumar and Sergey Levine. Model inversion networks for model-based optimization. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 5126–5137. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/373e4c5d8edfa8b74fd4b6791d0cf6dc-Paper.pdf.

Ruoshi Liu, Chengzhi Mao, Purva Tendulkar, Hao Wang, and Carl Vondrick. Landscape learning for neural network inversion, 2022. URL https://arxiv.org/abs/2206.09027.

Kaifeng Lyu and Jian Li. Gradient descent maximizes the margin of homogeneous neural networks, 2020. URL https://arxiv.org/abs/1906.05890.

Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them, 2014. URL https://arxiv.org/abs/1412.0035.

Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning, 2018. URL https://arxiv.org/abs/1805.04049.

Alexander Mordvintsev, Christopher Olah, and Mike Tyka. Inceptionism: Going deeper into neural networks, 2015. URL https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html.

Anh Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks, 2016. URL https://arxiv.org/abs/1605.09304.

Anh Nguyen, Jeff Clune, Yoshua Bengio, Alexey Dosovitskiy, and Jason Yosinski. Plug & play generative networks: Conditional iterative generation of images in latent space, 2017. URL https://arxiv.org/abs/1612.00005.

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 5, 03 2020. doi: 10.23915/distill.00024.001.

Emad W. Saad and Donald C. Wunsch. Neural network explanation using inversion. *Neural Networks*, 20(1):78–93, 2007. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2006.07.005. URL https://www.sciencedirect.com/science/article/pii/S0893608006001730.

Shibani Santurkar, Dimitris Tsipras, Brandon Tran, Andrew Ilyas, Logan Engstrom, and Aleksander Madry. Image synthesis with a single (robust) classifier, 2019. URL https://arxiv.org/abs/1906.09453.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

Pirzada Suhail. Network inversion of binarised neural nets. In *The Second Tiny Papers Track at ICLR 2024*, 2024. URL https://openreview.net/forum?id=zKcB0vb7qd.

Pirzada Suhail and Amit Sethi. Network inversion of convolutional neural nets. In *Muslims in ML Workshop co-located with NeurIPS 2024*, 2024. URL https://openreview.net/forum?id=f9sUu7U1Cp.

Pirzada Suhail and Amit Sethi. Network inversion for generating confidently classified counterfeits, 2025. URL https://arxiv.org/abs/2503.20187.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2014. URL https://arxiv.org/abs/1312.6199.

Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy, 2019. URL https://arxiv.org/abs/1805.12152.

Zihan Wang, Jason Lee, and Qi Lei. Reconstructing training data from model gradient, provably. In Francisco Ruiz, Jennifer Dy, and Jan-Willem van de Meent, editors, *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pages 6595–6612. PMLR, 25–27 Apr 2023. URL https://proceedings.mlr.press/v206/wang23g.html.

Eric Wong. Neural network inversion beyond gradient descent. In *WOML NIPS*, 2017. URL https://api.semanticscholar.org/CorpusID:208231247.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017. URL https://arxiv.org/abs/1708.07747.

Ziqi Yang, Jiyi Zhang, Ee-Chien Chang, and Zhenkai Liang. Neural network inversion in adversarial setting via background knowledge alignment. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, page 225–240, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450367479. doi: 10.1145/3319535.3354261. URL https://doi.org/10.1145/3319535.3354261.

Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization, 2015. URL https://arxiv.org/abs/1506.06579.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Not Applicable]
   (b) Complete proofs of all theoretical results. [Not Applicable]
   (c) Clear explanations of any assumptions. [Not Applicable]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes/No/Not Applicable]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Not Applicable]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]