Privacy Preserving Properties of Vision Classifiers: Vulnerability Analysis of Inversion Attacks for Training Data Reconstruction

Resnick Singh
Department of Electrical Engineering
Indian Institute of Technology Bombay
Mumbai, India
21d070055@iitb.ac.in

Abstract

Vision classifiers are now used in many privacy-sensitive applications. However, they remain vulnerable to attacks that can reconstruct their private training data. In this paper, we study how specific features of a trained neural network make it possible to recover those private samples. We explore two main approaches: A loss-based method that uses the mathematical conditions the network satisfies during training and A network-inversion method that relies on specially trained generators. Using the MNIST digit dataset, our experiments show that the choice of model architecture significantly impacts how easily data can be reconstructed. Simple Multi-Layer Perceptrons are far more vulnerable than Convolutional Neural Networks or Vision Transformers. We also compare different loss functions and find that the so-called stationary loss clearly captures the link between model parameters and training data. Overall, our findings suggest that choosing specific network designs can gain natural privacy benefits and make inversion attacks much harder.

1 Introduction

The use of vision classifiers is growing fast in areas like healthcare, finance, and biometrics. These models often work with very private data, which raises real worries about privacy. Even though these tools are very useful, they can accidentally reveal sensitive information through their settings (parameters). This risk needs careful attention. Laws such as the European Union's General Data Protection Regulation (GDPR) show how important it is to protect personal data in machine learning.

Most studies so far have looked at attacks like membership inference and model stealing. But rebuilding the actual training images or data from a model is an even bigger privacy problem. If someone can do that, it breaks the basic idea that sharing a model is safe.

In this work, we ask two main questions:

1. How much of the original training data can be rebuilt from a vision classifier's parameters?
2. Which features of these models make them easier or harder to attack in this way?

We build on ideas from Lyu & Li (2019) and Ji & Telgarsky (2020), and introduce practical ways to recover training data from a model.

Our key contributions are:

Preprint. Under review.

- A clear comparison of different reconstruction methods applied to vision classifiers, showing how well they work on various model designs.
- A study of different loss functions for reconstruction, with a focus on why "stationary loss" is so important.
- Real-world tests that prove a model's architecture can make it more or less vulnerable to data recovery.
- Simple guidelines for choosing model designs that keep privacy stronger against these inversion attacks.

2 Related Work

2.1 Network Inversion Techniques

Network inversion is a way to work backwards through a trained neural network, finding the kind of inputs that would produce certain outputs. In 2024, [5] introduced a method called "Network Inversion for Training-Like Data Reconstruction." They use a generator, conditioned on class labels, to learn what each class's input space looks like. Their approach takes advantage of how classifiers behave with their training data, and it also brings in expert knowledge about the kinds of inputs they expect. They showed that, by properly setting up the generator, you can recreate meaningful inputs from the network's learned features. Rather than trying to copy exact training samples, these methods focus on reversing the feature representations that the networks develop.

2.2 Training Data Reconstruction

In our work, we look at methods that try to rebuild the data used to train a model. Lyu and Li (2019) [4] and Ji and Telgarsky (2020) [3] showed that, under certain conditions, a neural network's parameters follow specific equations tied to the original training data. This idea led to optimization methods that recover training data from a trained model.

[1] used the above method and applied on Binary Classifier which was further extended by [2] to models that classify many classes. They turned the task into a constrained optimization problem: find inputs that meet the Karush–Kuhn–Tucker (KKT) conditions for the maximum-margin problem the network solves during training. But these methods assume the model has perfect accuracy on the training set and non-homogenous architecture (no skip connections).

3 Methodology

3.1 Vision Classifier Architecture

For our primary experiments, We used a simple Multi-Layer Perceptron (MLP) with:

- **Input layer:** 784 neurons (each 28×28 MNIST image flattened)
- **Hidden layers:** several fully connected layers with ReLU activation
- **Output layer:** 10 neurons (one for each digit 0–9)

We trained this architecture under two distinct scenarios:

1. **Full Training Set**
    - Used all 60,000 MNIST training images
    - Trained for 30 epochs
    - *Training accuracy:* 98.42%
    - *Validation accuracy:* 98.33%
2. **Small Training Set (Overfitting Case)**
    - Used only 600 training images
    - Trained for 30 epochs

- *Training accuracy:* 100%
- *Validation accuracy:* 87.64%

By comparing the two cases, we see how overfitting makes the model memorise its training data. A model that memorises too much can leak information about its samples, which is important when studying reconstruction attacks.

3.2 Reconstruction Techniques

3.2.1 Conditioned Generator

For the network inversion approach, we designed a conditioned generator that produces images given class labels. This generator learns the distribution of inputs that would activate specific classes in the target classifier. The architecture incorporates mechanisms to facilitate diverse yet realistic image generation.

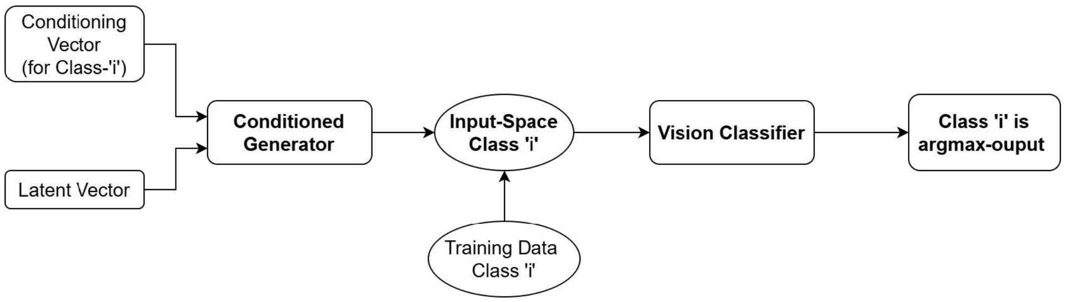

Figure 1: Conditioned Genrator

3.2.2 Loss Functions

We employ a combination of loss functions to guide the generator toward producing authentic training samples:

- **Cross-Entropy & Perturbed Cross-Entropy Loss:** Ensures generated samples receive high confidence scores from the target model, just like real training data.
- **Variational & Pixel Loss:** Constrains pixel values to lie between 0 and 1 and promotes smoothness, so the images remain realistic.
- **Gradient Loss:** Encourages the generator to produce samples that yield small model gradients, mimicking the behaviour of true training inputs.
- **Cosine & Orthogonality Loss:** Reduces feature similarity between generated samples and enforces orthogonal feature vectors to boost diversity.
- **Stationary Loss:** Leverages insights about convergence in gradient-based training, ensuring generated samples sit at stable points where further training changes very little.

$$L_{st}(x_1,...,x_m,\lambda_1,...,\lambda_m) = \|\theta - \sum_{i=1}^{m} \lambda_i \nabla_\theta [\Phi_{y_i}(x_i;\theta) - \max_{j \neq y_i} \Phi_j(x_i;\theta)]\|_2^2 \tag{1}$$

Where θ represents the model parameters, x_i are the inputs being reconstructed, λ_i are non-negative coefficients, and $\Phi_j(x_i;\theta)$ denotes the model's output for class j given input x_i.

This formulation derives from the KKT conditions of the maximum-margin problem implicitly solved during gradient-based training, as identified in prior theoretical work. The optimization process seeks to find inputs x_i and coefficients λ_i such that the stationary loss is minimized, effectively identifying points that could have been used in training the model.

4 Experiments and Results

4.1 Experimental Setup

We ran our tests on the MNIST dataset using the models and training settings described earlier. For each trained model, we applied our reconstruction methods with different loss function setups to see how well we could recover the original training images. We used three setups:

- All loss functions together with equal weights
- Only the stationary loss
- All loss functions except the stationary loss

This way, we could understand how much each loss component contributed to the final reconstruction quality.

4.2 Reconstruction Results

Our experiments revealed several key findings:

- **Stationary Loss Dominance.** When all loss functions were combined, the stationary loss tended to dominate the optimization process, likely due to its relatively higher magnitude compared to other components. This dominance suggests that the theoretical foundations of this loss effectively capture the relationship between model parameters and training data.
- **Architecture-Dependent Vulnerability.** Multi-layer perceptrons (MLPs) were much more open to reconstruction attacks than CNNs or Vision Transformers. That makes sense, because CNNs share weights and transformers use attention—both make it harder to work backwards from outputs to inputs.
- **Hyperparameter Sensitivity.** How well we reconstruct images changes a lot depending on the loss weights (the hyperparameters). Some weight settings give much clearer reconstructions, so picking the right numbers is very important.
- **Variation in Image Quality.** The look of the rebuilt images can be very different across settings. In the best runs, the reconstructed digits almost match the real training pictures, keeping the same style and structure.

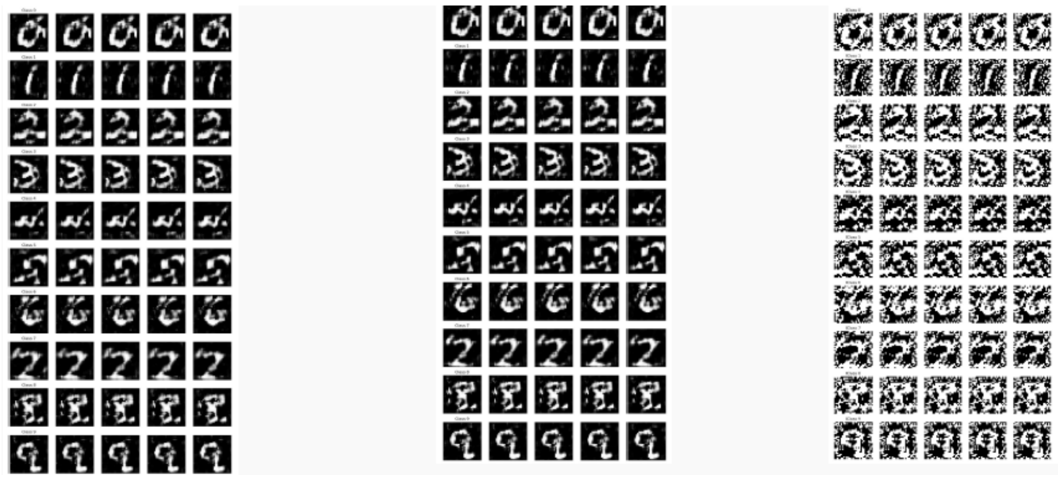

Figure 2: Comparison of reconstruction results with different loss function configurations. Left: All loss functions. Middle: Only stationary loss. Right: Without stationary loss.

5 Discussion and Analysis

5.1 Key Insights

When we use a “stationary loss” to try and rebuild training data, we get real proof of an idea people had before: the exact point where gradient-based training stops is linked to the shape of the data we trained on. In other words, when a model finishes training with gradient descent, its weights quietly store bits of the original samples. That creates a privacy risk.

We also saw that some model designs make it much harder to rebuild the data. For example, sharing weights across layers or using attention seems to hide the direct link between a training example and the model’s parameters. That gives a built-in privacy boost.

Finally, we found that if an attacker does not know the exact hyperparameters used in training, their reconstruction attempts turn out very badly. This means one simple defense is to pick training settings that are hard to guess. If attackers can’t match those settings, they can’t rebuild your data.

5.2 Limitations

Several limitations should be noted:

1. Our experiments focused primarily on the MNIST dataset, which features relatively simple grayscale images. The effectiveness of reconstruction techniques may differ substantially for more complex datasets with higher-dimensional images.
2. The homogeneous nature of our primary model architecture (without skip connections) satisfies theoretical assumptions that may not hold for many modern architectures.
3. We observed that the stationary loss approach becomes problematic when applied to CNN architectures, suggesting that the theoretical foundations require adaptation for models with more complex parameter structures.

6 Conclusion and Future Work

This study shows that image-classification models can be tricked into revealing the images they saw. When attackers use certain loss functions to guide their search, they get much better reconstructions. It also depends on the model’s design: simple MLPs are far easier to attack than CNNs or Vision Transformers.

Mixing different loss functions makes the attack harder to set up, but that same difficulty could help us build stronger models by making the attacker’s job tougher.

Future work should focus on:

1. **Advanced Defensive Techniques**: Developing specific countermeasures against the reconstruction attacks demonstrated in our research, potentially incorporating adversarial training or differential privacy approaches.
2. **Extension to Complex Architectures**: Verifying our results on more complex architectures including larger-scale transformers and vision-language models, with particular attention to understanding how reconstruction effectiveness changes in models with skip-connections or non-homogeneous structures.
3. **Hyperparameter Optimization**: Building methods that are less sensitive to hyperparameter choices, potentially incorporating automated approaches to hyperparameter optimization to make attacks more practical in real-world scenarios.

These directions will contribute to a deeper understanding of the privacy implications of sharing trained vision models and guide the development of more privacy-preserving approaches to model deployment.

References

[1] Niv Haim, Gal Vardi, Gilad Yehudai, michal Irani, and Ohad Shamir. Reconstructing training data from trained neural networks. 2022.

[2] Niv Haim, Gal Vardi, Gilad Yehudai, Ohad Shamir, and Michal Irani. Reconstructing training data from trained neural networks. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 22911–22924. Curran Associates, Inc., 2022.

[3] Ziwei Ji and Matus Telgarsky. Directional convergence and alignment in deep learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 17176–17186. Curran Associates, Inc., 2020.

[4] Kaifeng Lyu and Jian Li. Gradient descent maximizes the margin of homogeneous neural networks, 2020.

[5] Pirzada Suhail and Amit Sethi. Network inversion for training-like data reconstruction, 2024.