Bachelor's Data Science
Deep Learning Under Attack: Revealing Vulnerabilities through Shadow Reconstructions

Paul Thielen
1529854

Supervisors
Dr. Hannah Pinson
Aurélien Boland

Thesis

Eindhoven, June 2024

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Department of Mathematics and Computer Science

Contents

Abstract

Model inversion (MI) attacks are categorized into optimization approaches, which use gradient-based optimization and inversion via a secondary model. This study bridges the gap between these two methods, focusing on a gray-box setting where the model's architecture is known but not its weights. The focus is on using a transposed convolutional neural network (TCNN) as an inverse model to reconstruct images from a convolutional neural network's (CNN) output vectors. To address the gray-box setting, a shadow model is trained on the MNIST dataset to mimic the target model's behavior. The effect on image quality is tested based on the input combinations for the TCNN, which consist of the output vector, gradient reconstructions obtained from the gradient-based optimization approach, and activations from the shadow model's last linear layer. Results on the MNIST dataset show that the stand-alone gradient-based optimization reconstructions have an average SSIM of 0.2409 $\pm$ 0.0068 and an average MSE of 0.0672 $\pm$ 0.0021. Furthermore, the TCNN, acting as the inverse model of the target model, achieves an average SSIM of 0.2298 $\pm$ 0.0041 and an average MSE of 0.0966 $\pm$ 0.0011 when passing just the output vector through the model. The results also showed that from the different input combinations, using gradient reconstructions as the input for the TCNN significantly enhances image fidelity, achieving an average SSIM value of 0.3727 $\pm$ 0.0030 and an average MSE value of 0.0649 $\pm$ 0.0009. These reconstructions have a 54.73% higher SSIM value than the stand-alone gradient-based optimization reconstructions. The reconstructions have 62.19% higher SSIM values than the standard TCNN reconstructions, which only obtain the output vector as input. These findings highlight the benefit of utilizing both directions of MI attacks to obtain the highest quality reconstructions. The effects of defensive techniques like output vector rounding and truncation are also examined. Rounding has minimal impact, while truncation increases SSIM; it makes the generated images converge to the class average and lose the subtle features of a given target image. This study highlights the risks of data leakage in deep learning and emphasizes the need for robust defense mechanisms to protect sensitive information against MI attacks.

1 Introduction

The integration of machine learning, particularly deep learning, into various domains has revolutionized how we approach problem-solving and data analysis. With their sophisticated architectures, deep learning models have demonstrated unparalleled efficiency [14] in tasks ranging from natural language processing [9, 33, 34] to image recognition [6, 16, 23]. However, adopting these models in applications dealing with sensitive information has raised significant security concerns. A particularly alarming issue is the potential for deep learning models to inadvertently reveal information about their training data through their parameters [2, 28]. Such vulnerabilities expose these models to model inversion attacks, where attackers aim to reconstruct the original training samples, thereby compromising data privacy [10].

Model inversion (MI) involves extracting information about the training data by analyzing the model's predictions [35]. Research on the topic has split into two directions. The first direction is the optimization approach, where one inverts a model by using gradient-based optimization in the data space [10, 11, 18, 30]. The second direction inverts a model by training a second model that acts as the inverse of the original one [7, 20, 38, 35]. This work will utilize the inverse model and gradient-based optimization approach to obtain higher-fidelity reconstructions.

It is important to note that most MI attacks are in a white-box setup. White box modeling produces models whose structure is not hidden [1]. This means that the internal parameters and weights are known to the attacker. The concern for MI attacks has been amplified by the feasibility of reconstructions in a gray/black box setting [28]. A black-box setting is when the internal functioning of the model is hidden [1], and a gray-box setting is in between a white and black box setup. These approaches utilize "shadow models" that mimic the target model's behavior to facilitate the reconstruction of training data from output vectors [28]. These concepts underscore the urgent need for strategies to safeguard against such susceptibilities.

As mentioned earlier, this work will utilize the inverse model and gradient-based optimization approach to obtain higher-fidelity reconstructions. Additionally, the focus will be on a gray-box model setup to further highlight the dangers of MI attacks. The following assumptions are maintained throughout the work. The gray-box model setup within this work implies that the attacker knows the target model's architecture but not its weights, and the attacker has a disjoint dataset that is from the same distribution as the target model's training data. Furthermore, the target model will be a convolutional neural network (CNN). Lastly, reconstructions will be performed by passing output vectors from the CNN model as inputs through the inversion model, which is a transposed convolutional neural network (TCNN).

In machine learning, more input data is considered to improve model performance. Therefore, this work's scope will investigate how different input data combinations impact the fidelity of reconstructions. The first input for the inverse model is the output vector obtained from the CNN. The following inputs are computed using the output vector and the shadow model that imitates the target model behavior. The second input option is a flattened representation of a gradient-based reconstruction of a target image. The third input option are the last linear activations obtained from the shadow model's CNN. The activations are calculated using the trained shadow model's frozen weights. The ability to extract additional information (gradient-based reconstructions and last linear activations) from just the output vector and a shadow model in a non-white-box setup allows for the potential of reconstructing higher fidelity images.

The possibility of having the approach described above yielding high-fidelity images is concerning. Therefore, it will also be investigated how standard prevention techniques such as truncation and

output vector rounding impact the reconstruction quality of the proposed MI attack. By advancing our understanding of these vulnerabilities and potential defenses, we can create a more secure and trustworthy environment for the deployment of machine learning models [21].

1.1 Research Question

In light of these considerations, the central research question of this work emerges: How do different factors influence the fidelity of image reconstruction using TCNNs within a non-white box framework? Specifically:

- **RQ1:** How do variations in input combinations to the TCNN, including output vector, gradient reconstruction, and last linear layer activations, impact reconstruction quality?
- **RQ2:** How do defensive mechanisms, like truncation and output vector rounding, affect the quality of the reconstructed images?

2 Related Work

Earlier work explored model inversion attacks, where the goal was to reconstruct the input from a trained model's output. Fredrikson et al. [10] implemented a model inversion attack on a multi-layer perceptron (MLP) model to generate a recognizable image of a person using only their name or identifier known from the model. The main mechanism used was gradient descent, in which they optimized a cost function designed to reconstruct the input image as accurately as possible from the model's output [10]. It is important to note that their approach operated in a white-box setup, allowing access to the original model's parameters and weights.

Gradient-based reconstructions are not always the best choice. CNNs use max-pooling, which reduces the spatial dimensions of the input feature maps while retaining the most significant information (the maximum values in a pooling window) [36]. However, this gives rise to the coordinate transform problem. This problem is that the exact location of the maximum value is not retained. Without the retention of the maximum value location, subsequent layers lose the information about where features are located relative to each other within the original input space. Since localization is essential in reconstructions, gradient-based approaches may give rise to blur or distortion in reconstructions due to loss of spatial information during the gradient computation. It is important to note that this issue is specific to CNNs. Although not used in this work, Vision Transformers (ViTs) inherently handle spatial information through attention mechanisms and positional encodings, hence mitigating the spatial information loss seen in CNNs [8].

Zhao et al. [38] proposed a model inversion approach using transposed convolutional neural networks (TCNN) to reconstruct images effectively. Their architecture combines a CNN for feature extraction with a subsequent TCNN for image generation [38]. This CNN-TCNN structure is analogous to a generator network [37]. Notably, they utilized the Mean Squared Error (MSE) loss function to guide the image reconstruction process. The issue with MSE is that it is not normalized in representation [26] and treats all pixel errors equally without considering the structural or semantic importance of different parts of the image [19]. This means essential features like edges and textures, crucial for human perception, may not be accurately reconstructed. This downfall will be addressed by using the Structured Similarity Indexing Method (SSIM).

An advantage of TCNNs is their ability to retain spatial information during reconstruction. This is a marked improvement over gradient-based inversion methods, which often struggle to preserve spatial details [38]. TCNNs perform a procedure called "upsampling" by using transposed convolutional layers that efficiently learn a way of approximately undoing the pooling operations learned during CNN feature extraction. Progressive restoration of the image's spatial resolution thus allows for a more precise and sharp reconstruction.

The previous research has required access to the original model parameters or weights. To overcome the limitations of direct model access, the concept of shadow models will be utilized. Shadow models, as discussed by Shokri et al. [28], represent an innovative solution to perform membership inference without needing direct access to the original model's specifics. These models mimic the behavior of the original models by learning from datasets presumed to be similar to those used in the original training process [28]. Despite not being perfect replicas, shadow models provide a solution to not having the target model's weights and parameters. The concept of shadow models will be translated from the membership inference task to this work's task of image reconstruction.

3 Methodology

3.1 Shadow Model

The shadow model in this research setup learns from image data $D_T^S \subset D^S$ drawn from the same distribution as the data used to train the target model $D_T^T \subset D^T$. Here,D^S and D^T represent the entire datasets for the shadow and target models. Importantly, D^S and D^T are disjoint ($D^S \cap D^T = \emptyset$), ensuring no data leakage. This setup allows the shadow model to learn relevant features and patterns applicable to the target model's domain. Additionally, a validation set for the target model $D_V^T \subset D^T$ and a shadow model validation set $D_V^S \subset D^S$ monitor the training process and prevent overfitting, where $D_V^T \cap D_T^T = \emptyset$ and $D_V^S \cap D_T^T = \emptyset$. Comparing training loss with validation loss allows for effective hyperparameter tuning and addressing potential overfitting.

The shadow model's outputs are used to train the TCNN model. Why would one implement the shadow model if one could train the TCNN based on the target CNN's outputs? The answer lies in the nature of target models functioning in a black box setting, which means there is no access to their internal workings. Currently, developers of state-of-the-art AI systems often keep most details of their models private [3], making the black-box setting increasingly common.

Furthermore, most of the time, in a real-world scenario, an attacker does not have access to the training data of the target model. The reason for the training datasets not being publicly available is that the datasets are either proprietary or contain sensitive data that cannot be shared publicly due to privacy laws and confidentiality agreements [4, 5, 29]. A workaround to not having the exact training dataset used for the target model is to use data from the same distribution as the target model's training data. By training a shadow model on this data, one can gain full access to a surrogate (the shadow model) of the target model from which one can extract valuable information, such as the last activation layer discussed in Section 3.5.

3.1.1 Shadow & Target Model Architecture

For this research, a simple CNN architecture is implemented for the shadow model denoted as M_S. This architecture aligns with the target model, denoted as M_T, and is well-suited for capturing spatial features in image data. The CNN architecture consists of three convolutional blocks, each following the same structure:

1. A convolutional layer, tasked with extracting feature maps from the input images.
2. A batch normalization layer, which normalizes the activations of the previous layer, thereby enhancing the stability and speed of the network's training phase.
3. A max-pooling layer reduces the spatial dimensions of the feature maps, thus mitigating the risk of overfitting and enhancing the network's ability to generalize.
4. A Rectified Linear Unit (ReLU) activation layer is responsible for introducing non-linearity into the network, which allows it to learn more complex patterns in the data.

Two fully connected layers are added after the final convolutional block to learn more complex relationships between the extracted features. The architecture is visually represented in Figure 1. To obtain the optimal model architecture that balances accuracy and robustness, the combinations of dropout and activation functions between the linear layers are investigated in Section 4.2. The final CNN architecture details for M_S and M_T can be found in Appendix A.

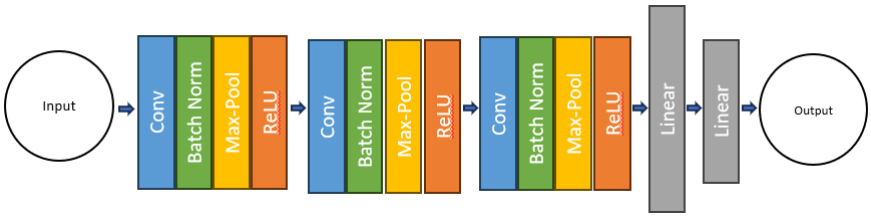


Figure 1: Architecture of the Convolutional Neural Network (CNN)

3.2 TCNN

The primary goal of implementing a TCNN within this study is to obtain higher-quality image reconstructions that surpass the capabilities of gradient-based reconstructions from a CNN. TCNNs ability to retain spatial information during reconstruction makes it a better alternative in comparison to a gradient-based inversion method, which often struggles to preserve spatial details [38].

3.2.1 TCNN Architecture

For this research, a simple TCNN architecture has been developed and implemented. The architecture comprises of three distinct transposed convolutional blocks aimed at reconstructing the original image from an output vector. Each block follows the following structure:

1. A transposed convolutional layer. This layer performs an approximate inverse operation of a typical convolutional layer. Transposed convolutional layers increase the spatial dimensions of the output, which is helpful for tasks such as image segmentation and generating high-resolution images from low-resolution inputs.
2. A batch normalization layer. Following the transposed convolutional layer, the batch normalization layer normalizes the activations. This step is vital for enhancing the stability and speed of the network's training phase by reducing internal covariate shift, where the distribution of layer inputs changes as the parameters of the previous layers change.
3. A ReLU activation layer. This layer introduces non-linearity to the network, enabling it to learn more complex patterns in the data. The ReLU function is employed here, which outputs the input directly if it is positive; otherwise, it outputs zero.

Deploying this architecture, visualized in Figure 2, as the inversion model aims to increase the reconstruction quality of MI attacks. Using transposed convolutional layers effectively addresses and overcomes the limitations of conventional gradient-based approaches. This improved method is expected to bring about higher fidelity reconstructions than gradient-based reconstructions.

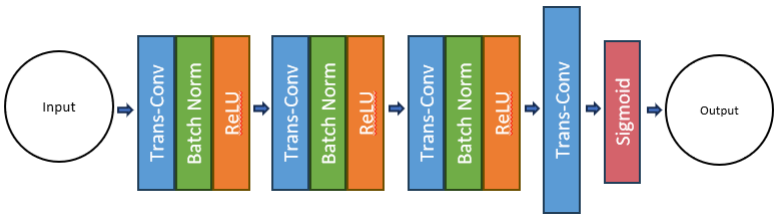


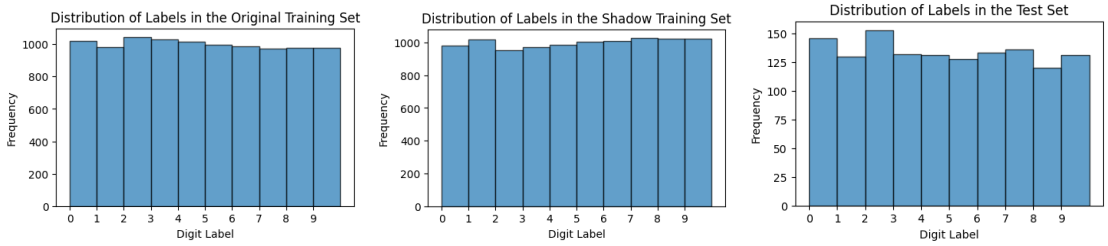
Figure 2: Architecture of the Transposed Convolutional Neural Network (TCNN)

3.3 Dataset

The Modified National Institute of Standards and Technology (MNIST) dataset is a well-established benchmark for image classification tasks, particularly for handwritten digits [15]. It consists of 70,000 grayscale images (28x28 pixels) labeled with digits 0-9. 60,000 are training images, and the remaining 10,000 are testing images. MNIST offers several advantages. Due to its simplicity, small size, and well-defined categories, MNIST is a great starting point for developing and evaluating image classification algorithms. Additionally, MNIST's widespread use allows for easy performance comparison between different models and approaches. Finally, the free and public availability of MNIST facilitates research reproducibility.

3.3.1 Preprocessing

For computational running time, a subset of 2,000 images per class (ie. 20,000 total images) would be taken from the 60k training images, and a subset of 400 images per class (ie. 4,000 total images) would be taken from the 10k testing images. The 20k training and 4k testing images were then split further down as this research requires a train and validation set for the target model, a disjoint but from the same distribution training and validation set for the shadow model, and finally, a test set is required (D^F). The 20k training images were randomly split into a 50/50 ratio for the target and shadow train sets, D_T^T, D_T^S respectively. The distribution per class can be seen in Figure 3a and 3b, respectively. The 4k test set was split randomly into 3 equal but disjoint parts to create the target validation set D_V^T, shadow validation set D_V^S, and test set D_F. Figure 3c shows the test data distribution. All the images were reshaped from 28×28 to 32×32 as seen in [35, 38] to fit the model architecture. Figure 3 shows that there is no under or over-representation of any given class within the different datasets. This is important because machine learning models frequently exhibit drops in performance under the presence of distribution shifts [22].



(a) Target Train Set (D_T^T) (b) Shadow Train Set (D_T^S) (c) Test Set (D_F)

Figure 3: Class distributions of datasets

3.4 Gradient Reconstructions

As mentioned in Section 2, gradient reconstructions have been used in an MLP setting, and this work will extend on this and construct reconstructions from a CNN model's outputs. These reconstructions will then be flattened from a 32×32 image into a 1×1084 vector and passed as such into the TCNN model to obtain even more realistic reconstructions.

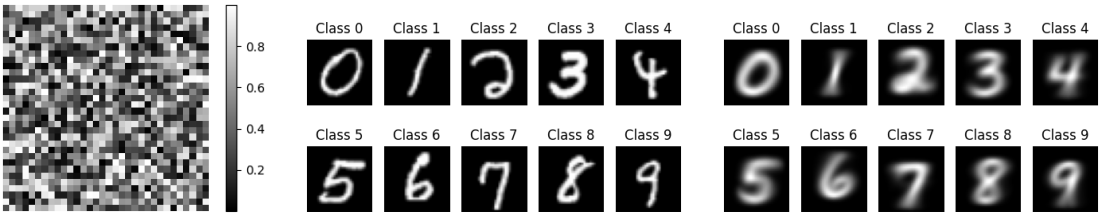
The image to be optimized I can be initialized in 3 different ways: randomly shown in figure 4a, from a specific instance of a class shown in figure 4b, or from the class average shown in figure 4c. Class averages and instances are taken from the shadow model training data D_T^S, which, as previously

mentioned, is disjoint from the target model's CNN training data. There are benefits and drawbacks to each type of initialization.

Random initialization of image I provides an unbiased starting point and generalizability as it does not provide any prior knowledge about the class and could potentially lead to discovering more generalized features. The downside is that the final reconstruction is susceptible to getting stuck in a local minima, and the number of iterations required to converge may be higher.

The reason for using a specific class instance initialization for image I is that it allows for higher fidelity and faster convergence. Since one starts from an actual class instance, the gradient updates will refine a real image rather than one from random noise and require fewer iterations, as the initial point should be close to the target reconstruction. The downside is that a specific instance might bias the reconstruction towards specific features of that particular class. Furthermore, if the specific instance selected from a class is not representative of that class, it can lead to reconstructions that are part of the given class but cannot deviate from the unique features of the initial image.

The final initialization method uses class averages for the image I. This technique helps achieve more stable convergence by avoiding the outlier characteristics of a class found in individual class examples. As a result, it can recreate the most representative features of the instance with fewer iterations. However, the reconstruction might focus on irrelevant features if the class average is not well aligned with the class characteristics relevant to the output vector.



(a) Random Noise (b) Specific Class Instance (c) Class Average

Figure 4: Different types of initializations of image I at $t = 0$

A few definitions have to be made to calculate the gradient-based reconstructions. Let I be a set of initialized images that will be updated at each time step t to converge to the corresponding target images. Within this work, I will be updated 400 times. y represents the target labels of I. Furthermore, let M_S be the shadow CNN model.

$$I_t = I_{t-1} - \eta \cdot \nabla_I L_{\text{total}}(I_{t-1}),$$
$$I_t = \text{clamp}(I_t, 0, 1) = max(0,\ min(I_t, 1))$$

Images I are updated based on the gradient of the total loss, and their values are clamped to ensure they remain within valid image pixel ranges (0 to 1). The loss function used is defined below:

$$L_{\text{total}}(I, y) = L_{\text{CE}}(I, y) + R_{L2}(I) + R_{\text{TV}}(I) \tag{1}$$

The combination of cross-entropy loss (L_{CE}), L_2 regularization (R_{L2}), and total variation regularization (R_{TV}) results in a comprehensive overall loss function that balances classification accuracy, perturbation magnitude, and visual coherence. Each of the loss function's components serves a very

distinct and vital function to optimize and enhance the reconstruction of the optimized image.

$$L_{\text{CE}}(I, y) = -\sum_{c=1}^{C} y_c \log(\text{softmax}(M_S(I))_c) \tag{2}$$

$$R_{L2}(I) = \lambda_{\text{reg-l2}} \sum_{i,j} I_{i,j}^2 \tag{3}$$

$$R_{\text{TV}}(I) = \lambda_{\text{reg-tv}} \sum_{i,j} (\|I_{i,j} - I_{i+1,j}\| + \|I_{i,j} - I_{i,j+1}\|) \tag{4}$$

Within Formula 2, C represents the total number of classes in the classification problem (MNIST: $C = 10$). Here, y_c is either 0 or 1, indicating whether the class label is the correct classification. $M_S(I)$ represents passing image I through the model M_S. Formulas 3 and 4 use $I_{i,j}$ notation; within this work, i, j represent the indices of the image I. Lastly $\lambda_{\text{reg-l2}}$ and $\lambda_{\text{reg-tv}}$ are the two hyper-parameters that will be tuned in section 5.1.

Cross-entropy loss (L_{CE}) measures the discrepancy between the predicted labels and the true labels. It effectively guides the model towards more accurate predictions by penalizing deviations from the true class labels [17]. L_2 regularization (R_{L2}) and Total Variation regularization (R_{TV}) are both used to impose smoothness in optimization problems, but they do so in fundamentally different ways. R_{L2} penalizes large values in I to ensure smaller perturbations, and it encourages the image to have a globally low dynamic range, making it uniformly smooth without necessarily considering the relationships between adjacent pixels [13]. R_{TV} targets the spatial variation between adjacent pixels. It penalizes the sum of the absolute differences between neighboring pixel values. This approach encourages spatial coherence by making the value of a pixel close to that of its neighbors. Overall, it encourages local smoothness while still allowing for sharp transitions, which is crucial in maintaining edge integrity in images [24].

3.5 Linear Activations

The classifier model (CNN) utilizes two linear layers to go from the last feature map to the output vector. The MNIST dataset's output vector is of shape ten as there are ten classes. In this work, it will be tested if a TCNN performs better in the task of reconstructing images if more information is passed as input. Previous work by Zhao et al. [38] used alongside the output vector, the flattened gradient-weighted class activation mapping, known as Grad-CAM, of the input to enhance the reconstruction quality. Grad-CAMs use the gradients of any target concept, flowing into the final convolutional layer to produce a coarse localization map highlighting important regions in the image for predicting the concept [27]. Grad-CAMs are 2D, but Zhao et al. [38] turned them into 1D arrays to concatenate them with output vectors. They saw significant improvement in their reconstructions due to the added information. Within this work, Grad-CAMs will not be utilized as grad-cams require access to the gradients with respect to specific layers within the model [27]. As access to the target model is restricted and passing the target model's image set D^T through the shadow model breaks the rules of the setup, calculating the Grad-CAMs for the output vectors from the target model proved impossible. Instead, the activations from the shadow model's last linear layer will be extracted, resulting in 50 additional values that can be utilized in the reconstruction process. Although the last linear activations are not the same as Grad-CAMs, the aim is still to see if the additional input to the TCNN will help in achieving higher-fidelity reconstructions.

To calculate the activations of the shadow model's last linear layer, the following formulas are used:

$$A_t^L = \begin{cases} A_{t-1}^L - \eta \nabla_{A_{t-1}^L} L & \text{if } L(A_{t-1}^L) > \tau \\ A_{t-1}^L & \text{otherwise} \end{cases}$$

$$\mathcal{L} = MSE(Z_t, Y) = \frac{1}{n}\|Z_t - Y\|^2 \text{ where } Z_t = A_{t-1}^L W^T + b$$

From the trained shadow model, M_S, the last linear layer weights (W) and bias (b) are extracted. Z_t is the result of a linear transformation of the activation layer A_t^L by the weights W followed by the addition of bias b. The goal is to find A_t^L so that the MSE loss between Z_t and the target output Y is minimized. The activation layer A_t^L is updated by taking the previous state of the activation layer and subtracting a term proportional to the gradient of the loss function $\mathcal{L}$ with respect to A_{t-1}^L multiplied by the learning rate η. Due to the iterative updates to A^L, there is a set condition that A^L shall continue to update for the current batch of output vectors until the loss falls below the threshold of τ (set at 1.5).

3.6 Multi-Modal Inversion Attack Model

The baseline approach for training the TCNN involves using the output vector as the input. However, other input variations may enhance image reconstruction. As previously stated, [38] demonstrated that incorporating a flattened grad-cam into the input can effectively transform the TCNN into a multi-modal system, thereby improving reconstruction quality. This work aims to delve deeper into multi-modal inversion by exploring the integration of additional variables.

Specifically, this work examines the extraction of gradient reconstructions and the linear activations from the last linear layer of a shadow CNN, as discussed earlier. The analysis focuses on how the different input combinations of the output vector, the flattened gradient reconstruction, and the last linear layer activations affect the fidelity and precision of the reconstructions. This investigation will contribute to a more comprehensive understanding of how different input modalities influence the performance of reconstructions with the use of TCNNs. By training and evaluating the TCNNs and comparing the SSIM and MSE values (refer to Section 4.1.2), the study aims to determine the input configuration that maximizes the quality of image reconstructions.

3.7 Workflow

As multiple components have been discussed up to this point, this section aims to provide an overview of how each previously discussed component interacts with each other and flows through the training and testing procedure.

3.7.1 Training Procedure

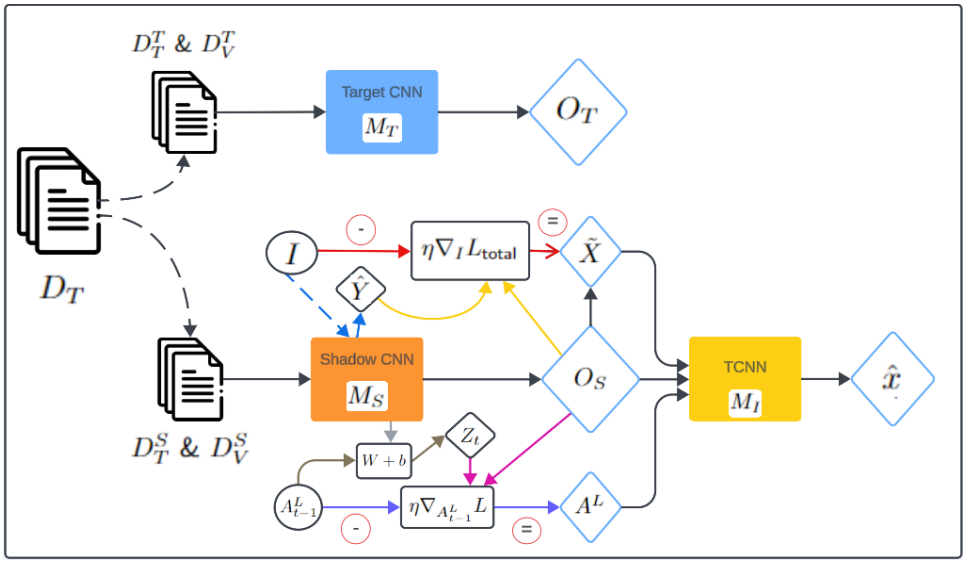


Figure 5: This diagram illustrates the training procedure involving three key components: the Target CNN (M_T), Shadow CNN (M_S), and TCNN (M_I).

Training can be broken down into three parts: the training of the target CNN model (M_T), the shadow CNN model (M_S), and the TCNN (M_I). The training data, D_T, is divided into target train D_T^T, target validation D_V^T, shadow train D_T^S, and shadow validation D_V^S sets (see section 3.3.1). M_T and M_S are trained on their respective training and validation data to obtain the output vectors O_T and O_S, respectively. Two more components are required to train the TCNN M_I.

Both gradient reconstructions and linear activations were described earlier in this work. For the gradient reconstructions, there is an image I that is initialized at $t = 0$ (reference Figure 4). I is then passed through M_S to obtain $\hat{Y}$ (blue arrows). Image I is updated based on the gradient of the total loss (see Formula 1). This process is repeated 400 times, at which point the gradient-based reconstruction $\tilde{X}$ is obtained (red arrows).

The linear activations A^L are updated based on the gradient of the MSE loss between Z_t and O_S (purple arrows). Z_t is obtained by multiplying the last linear activations A_{t-1}^L by the extracted weight matrix W and b (gray arrow) from the last fully connected layer from M_S (brown arrows). As A^L will be iteratively updated, there is a set condition that A^L shall continue to update until the loss falls below a set threshold. Refer to section 3.5 to review the process.

The basic training of M_I consists of having only the output vectors of the CNN as input (O_S). This work also explores how the different variations of input combinations to M_I impact reconstruction quality, hence the reason for calculating A^L and $\tilde{X}$. The output of M_I is $\hat{x}$, the final reconstructed image.

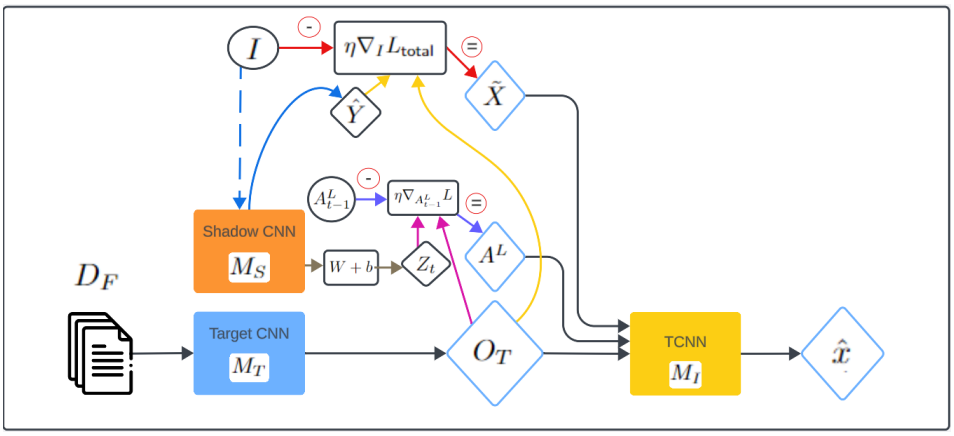


Figure 6: This diagram illustrates the testing procedure involving three key components: the Target CNN (M_T), Shadow CNN (M_S), and TCNN (M_I).

The testing procedure involves passing the test data D_F through the target CNN model M_T to obtain the output vectors O_T. To obtain the reconstructions from the TCNN M_I, the gradient reconstructions $\tilde{X}$ and the last linear activations A^L must be computed.

As this work outlines, access to M_T is restricted, so the trained shadow CNN model M_S will be used to obtain $\tilde{X}$ and A^L. The only aspect that changes in the calculations is that instead of using the output vector of the shadow model O_S, the output vector O_T is now used since the goal is to reconstruct the images from the output vectors from the target CNN model M_T. Afterward, the combinations of the output vector O_S, gradient reconstruction $\tilde{X}$, and last linear activations A^L are passed into the trained TCNN M_I to obtain the reconstructions of the test data $\hat{x}$.

4 Experimental Setup

4.1 Evaluation Metrics

4.1.1 Classification

The model that is under attack is a CNN model trained on the MNIST dataset. To evaluate how the CNN model is performing, the F1 score will be used. The reason is that the F1 score is balanced in terms of precision and recall. As the MNIST dataset contains ten classes, the weighted F1 scores will be used, which can be defined as:

$$F1_{\text{weighted}} = \sum_{i=1}^{n} \frac{N_i}{N} \cdot F1_i \tag{5}$$

In this formula, n is the total number of classes (in this case, 10). N_i is the number of true instances for class i. N is the total number of instances across all classes. $F1_i$ is the F1 score for class i, calculated as:

$$F1_i = 2 \cdot \frac{\text{Precision}_i \cdot \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i}$$

Precision (Precision_i) is defined as $\frac{TP_i}{TP_i+FP_i}$, and recall (Recall_i) is defined as $\frac{TP_i}{TP_i+FN_i}$. Here, TP_i represents the number of true positives, FP_i represents the number of false positives, and FN_i represents the number of false negatives for class i.

4.1.2 Image Reconstruction

The effectiveness of reconstructions is evaluated by the resemblance to the original input. To evaluate a reconstruction that is recognizable to a human, this research will leverage various aspects of reconstruction quality.

The Human visual perception system is highly capable of identifying structural information from a scene, which allows humans to identify the differences between the information extracted from a reference and a sample scene [32]. Unlike traditional metrics like Mean Squared Error (MSE) and Peak Signal-to-Noise Ratio (PSNR), which focus solely on pixel-wise intensity differences, the Structural Similarity Index (SSIM) offers a more comprehensive evaluation. It accomplishes this by incorporating considerations of local image structure, luminance, and contrast. This alignment with human visual perception makes SSIM a valuable tool, as it assesses the quality of reconstructed images from a perspective that resonates with humans.

This work, like Zhao et al. [38], will utilize 11×11 Gaussian kernels within the SSIM calculation to compare the different levels of granularity between two images. They calibrated τ to match the human perceived similarity such that $\tau = 1.5$ for the MNIST dataset. N is the total number of elements in the window (for an 11×11 window, $N = 121$).

$$w_i = \frac{\exp\left(-\frac{i^2}{2\tau^2}\right)}{\sum_{i=1}^{N} \exp\left(-\frac{i^2}{2\tau^2}\right)} \tag{6}$$

To delve deeper into the SSIM metric, image characteristics can be broken into three key components:

- **Luminance:** Measured by the mean intensity of the pixels. luminance, denoted as μ_x, reflects the overall brightness level. SSIM compares the luminance of the reconstructed image with the target image, denoted as $l(\mathbf{x}, \mathbf{y})$, to gauge their similarity in overall brightness.

$$\mu_x = \frac{1}{N} \sum_{i=1}^{N} w_i x_i \tag{7}$$

$$l(\mathbf{x}, \mathbf{y}) = \frac{2\mu_x \mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \tag{8}$$

Formula 7: x_i is the i'th pixel value of the image x. A Gaussian window (see Formula 6), w_i, focuses the overall SSIM calculation on the local region of the image. N is the total number of pixel values in image x.
Formula 8: x is the reconstructed image and y is the target image. $C_1 = (K_1 L)^2$. K is a small constant, and L is the dynamic range of the pixel values (0-1 for scaled images).

- **Contrast:** The standard deviation of pixel intensities captures the contrast level within an image. SSIM analyzes the contrast between the reconstructed and target images, ensuring the reconstructed image retains the same level of detail and variation in brightness as the original.

$$\sigma_x = \left(\frac{1}{N-1} \sum_{i=1}^{N} w_i (x_i - \mu_x)^2 \right)^{\frac{1}{2}} \tag{9}$$

$$c(\mathbf{x}, \mathbf{y}) = \frac{2\sigma_x \sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \tag{10}$$

Formula 9: x_i is the i'th pixel value of the image x. The same Gaussian window w_i is utilized as in Formula 7. N is the total number of pixel values in image x. μ_x is defined in Formula 7.
Formula 10: σ denotes the standard deviation of a given image. x is the reconstructed image and y is the target image. $C_2 = (K_2 L)^2$.

- **Structure:** This crucial aspect goes beyond mere intensity values and delves into the spatial arrangement of pixels. SSIM calculates the correlation coefficient between the corresponding regions of the reconstructed and target images. A high correlation signifies that the structural patterns and relationships between pixels in both images are well-preserved.

$$s_x = \frac{\mathbf{x} - \mu_x}{\sigma_x} \tag{11}$$

$$s(\mathbf{x}, \mathbf{y}) = \frac{\sigma_{xy} + C_3}{\sigma_x \sigma_y + C_3} \text{ where } \sigma_{xy} = \frac{1}{N-1} \sum_{i=1}^{N} w_i (x_i - \mu_x)(y_i - \mu_y) \tag{12}$$

Formula 11: x is a image. μ_x represents the mean of a given image and σ denotes the standard deviation of a given image.
Formula 12: σ denotes the standard deviation of a given image. x being the reconstructed image and y being the target image. $C_3 = (K_3 L)^2$.

The three components are then merged to form the complete SSIM formula. It is important to note that C_1, C_2, and C_3 are used to avoid instability within the equations. To simplify the formulas, the weights are set equal to each, $\alpha = \beta = \gamma$, and declare that $C_3 = C_2/2$. Additionally, L=1 as pixel

values in this work will range from 0-1, $K_1 = 0.01$, and $K_2 = 0.03$ [31].

$$\text{SSIM}(\mathbf{x}, \mathbf{y}) = [l(\mathbf{x}, \mathbf{y})]^{\alpha} \cdot [c(\mathbf{x}, \mathbf{y})]^{\beta} \cdot [s(\mathbf{x}, \mathbf{y})]^{\gamma} \tag{13}$$

$$\text{SSIM}(\mathbf{x}, \mathbf{y}) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{14}$$

To understand further details behind the math, see [31]. SSIM's utilization of luminance, contrast, and structure provides a more holistic evaluation of reconstructions. It ensures that the shadow model captures not only the overall intensity levels but also the intricate details and structural relationships within the images. This assessment is crucial for determining the effectiveness of the shadow model in replicating the target model's behavior. SSIM values range from -1 to 1, but in this work, the SSIM values are normalized to a range of 0 to 1, where:

- SSIM $= 1$ represents a perfect match between the reconstructed and target images. At this value, the reconstructed image's luminance, contrast, and structure are identical to the target image's, indicating flawless preservation of image quality and structural integrity.
- SSIM $= 0$ signifies no structural similarity between the reconstructed and target images. At this extreme, the reconstructed image bears no resemblance to the target image in terms of luminance, contrast, and spatial arrangement of pixels, indicating a complete loss of quality and structural fidelity.

MSE, despite its focus on pixel-wise intensity differences, is a fundamental and widely understood metric in image processing. Its mathematical simplicity and straightforward interpretation, as it measures the average squared difference between corresponding pixel values, provides a clear and objective measure of error. Therefore, both the SSIM and MSE will be reported in this research. Formula 15 illustrates how to calculate the Mean Squared Error (MSE) between two images, I_1 and I_2. Here, M and N are the dimensions of the images, and $I_1(i,j)$ and $I_2(i,j)$ represent the pixel values at position (i,j) in each image, respectively.

$$\text{MSE}(I_1, I_2) = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} (I_1(i,j) - I_2(i,j))^2 \tag{15}$$

4.2 CNN Training

To find the best architecture for the target CNN M_T, it was investigated whether adding an activation function or dropout between the last two fully connected layers would improve performance. The Rectified Linear Unit (ReLU), Sigmoid, Gaussian Error Linear Unit (GELU), and Sigmoid Linear Unit (SiLU) activation functions were tested. The probability of dropout was set at 50%.

The CNN model, M_T, was trained each time with D_T^T, and the validation set D_V^T is used to report the weighted f1 scores (refer to Formula 5). Each setup was trained and evaluated five times, and the respective weighted f1 score on the validation set, training loss, and validation loss are reported in Table 1. The loss function used to train the CNN models is cross-entropy loss (refer to Appendix A). As seen in Table 1, the best architecture for M_T is no activation function and utilizing a dropout of 50% between the two fully connected layers during training.

The target CNN model, M_T, and the Shadow model, M_S, utilize the best architecture found above and are trained on D_T^T and D_T^S, respectively. Both models are trained on 20 epochs and use their respective validation sets D_V^T and D_V^S, as described in Section 3.3.1. The weighted f1 score from the respective validation sets, training loss, and validation loss for both models can be found in Table 2.

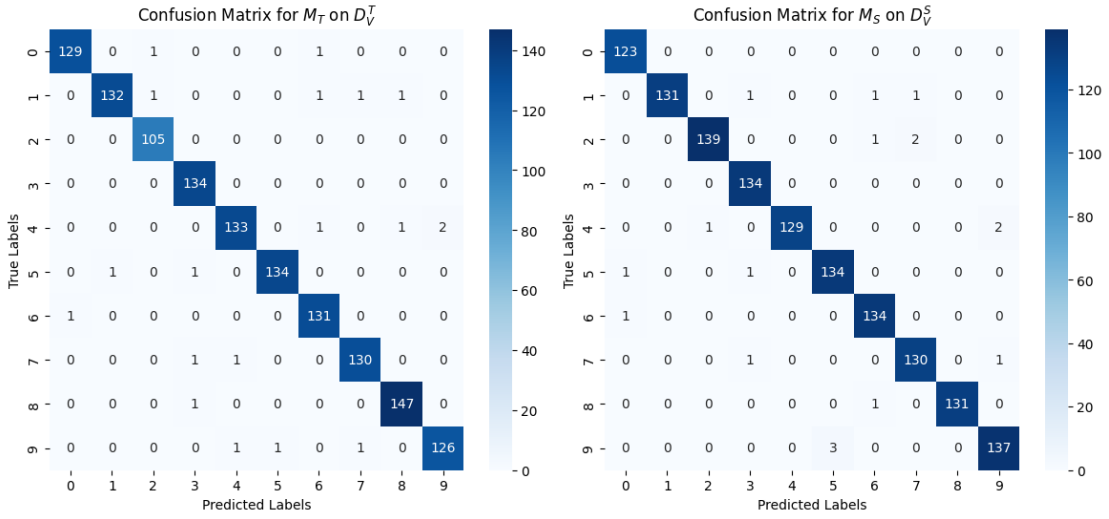
Activation Function	Dropout (p=0.5)	F1 Score	Training Loss	Validation Loss
None	No	0.9838 ± 0.0027	0.0179 ± 0.0108	0.0693 ± 0.0153
None	**Yes**	**0.9865 ± 0.0019**	0.0186 ± 0.0137	0.0585 ± 0.0149
ReLU	No	0.9750 ± 0.0114	0.0124 ± 0.0168	0.1308 ± 0.0727
ReLU	Yes	0.7113 ± 0.3255	1.4131 ± 0.4051	0.7747 ± 0.6425
Sigmoid	No	0.9851 ± 0.0030	0.0148 ± 0.0153	0.0533 ± 0.0182
Sigmoid	Yes	0.8829 ± 0.1553	0.6599 ± 0.3926	0.2994 ± 0.2636
GELU	No	0.9826 ± 0.0061	0.0075 ± 0.0165	0.0735 ± 0.0285
GELU	Yes	0.8918 ± 0.0909	1.3403 ± 0.2281	0.5522 ± 0.2578
SiLU	No	0.9770 ± 0.0081	0.0110 ± 0.0153	0.1062 ± 0.0402
SiLU	Yes	0.8714 ± 0.1602	1.2120 ± 0.3972	0.5598 ± 0.4410

Table 1: Impact of Dropout and Activation Functions on Fully Connected Layers in CNN Architecture

Furthermore, the confusion matrix for M_T and M_S can be found in figures 7a and 7b, respectively. Both the table and the matrices confirm that the shadow model is able to imitate the behavior of the target model while being trained on data from the same distribution but being disjoint ($D^S \cap D^T = \emptyset$) as mentioned in Section 3.1.

Model	F1 Score	Training Loss	Validation Loss
M_T	0.9856	0.0187	0.0483
M_S	0.9866	0.0288	0.0491

Table 2: Training & Validation loss of Target CNN (M_T) and Shadow CNN (M_S)



(a) Confusion Matrix from M_T

(b) Confusion Matrix from M_S

Figure 7: Confusion Matrices

5 Experiment Results

5.1 Hyperparameter Tuning for Gradient Reconstructions

With gradient reconstructions, three hyperparameters can be tuned. The first two hyperparameters are $\lambda_{\text{reg-l2}}$ and $\lambda_{\text{reg-tv}}$ seen in formulas 3, and 4 respectively. They affect the total loss function that is minimized in the process of gradient reconstructions (formula 1). The last hyperparameter investigated is the initialization of image I at $t = 0$, denoted as I_0, which can be random noise, a specific class instance, or a class average image.

To tune $\lambda_{\text{reg-l2}}$ and $\lambda_{\text{reg-tv}}$, random grid search was implemented. Random grid search is a method of hyperparameter tuning where, instead of exhaustively searching all possible combinations, it samples a subset of combinations at random. The search space for $\lambda_{\text{reg-l2}}$ is $\{\lambda_{\text{reg-l2}} \in \mathbb{R} \mid 1 \times 10^{-5} \leq \lambda_{\text{reg-l2}} \leq 3 \times 10^{-2}\}$, and the search space for $\lambda_{\text{reg-tv}}$ is $\{\lambda_{\text{reg-tv}} \in \mathbb{R} \mid 1 \times 10^{-6} \leq \lambda_{\text{reg-tv}} \leq 3 \times 10^{-2}\}$. The values were sampled from a log distribution from the specified ranges.

Sampling from a log-uniform distribution means generating random values from a distribution where the logarithm of the variable follows a uniform distribution. In other words, if X follows a log-uniform distribution, $log(X)$ will follow a uniform distribution. Mathematically, if X is log-uniformly distributed in the range $[a, b]$, it implies that $log(X)$ is uniformly distributed in the range $[log(a), log(b)]$.

Log-uniform distributions are commonly used in hyperparameter tuning for machine learning models. Parameters like learning rates and regularization strengths have a wide range of reasonable values that span multiple orders of magnitude. A log-uniform distribution allows for a broader search space that effectively covers small and large values.

The random grid search is broken down into the coarse search and the fine search. First, the coarse random search is performed, which considers the entire search space defined above for $\lambda_{\text{reg-l2}}$ and $\lambda_{\text{reg-tv}}$. This involves 20 iterations of random selection within the search spaces, with each iteration comprising 320 gradient reconstructions using the selected values for $\lambda_{\text{reg-l2}}$ and $\lambda_{\text{reg-tv}}$. The average SSIM of the reconstructions is saved. The gradient reconstructions are calculated from output vectors O_S, which are obtained by passing training images $I \subset D_T^S$ through the trained shadow model M_S. After the coarse random search is completed, the fine random grid search is performed to refine the hyperparameter values.

The fine search focuses on the most promising regions identified during the coarse search, allowing for a more precise determination of the optimal hyperparameters. Given the best-found values for $\lambda_{\text{reg-l2}}$ and $\lambda_{\text{reg-tv}}$ are $\lambda_{\text{best-l2}}$ and $\lambda_{\text{best-tv}}$ respectively, the fine search spaces are defined as $\{\lambda_{\text{reg-l2}} \in \mathbb{R} \mid 0.5 \cdot \lambda_{\text{best-l2}} \leq \lambda_{\text{reg-l2}} \leq 1.5 \cdot \lambda_{\text{best-l2}}\}$ and $\{\lambda_{\text{reg-tv}} \in \mathbb{R} \mid 0.5 \cdot \lambda_{\text{best-tv}} \leq \lambda_{\text{reg-tv}} \leq 1.5 \cdot \lambda_{\text{best-tv}}\}$. The fine random grid search consists of 10 iterations within the fine search spaces, with each iteration comprising 320 gradient reconstructions using the selected values for $\lambda_{\text{reg-l2}}$ and $\lambda_{\text{reg-tv}}$. The average SSIM of the reconstructions is saved. Just like in the coarse random search, the gradient reconstructions are calculated from output vectors O_S, which are obtained by passing training images $I \subset D_T^S$ through the trained shadow model M_S. Figure 8 shows the average SSIM values for $\lambda_{\text{reg-l2}}$ and $\lambda_{\text{reg-tv}}$. As the goal is to maximize SSIM values, the best hyperparameters are: $\lambda_{\text{reg-l2}} \approx 4.4323 \times 10^{-5}$ and $\lambda_{\text{reg-tv}} \approx 9.487\,85 \times 10^{-3}$

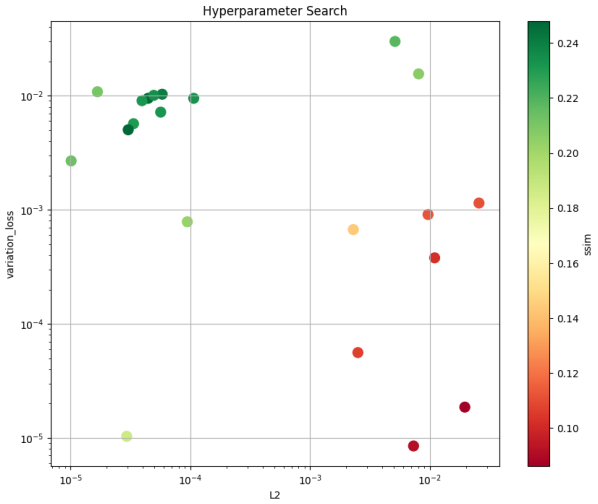


Figure 8: Random Grid Search for $\lambda_{\text{reg-l2}}$ and $\lambda_{\text{reg-tv}}$

To investigate which initialization method is the best, gradient reconstructions were done for all images in the validation set D_V^S, in which the target output vector is obtained from the shadow model M_S, and then I_0 is optimized to minimize the loss. Furthermore, the gradient reconstructions were done for all images in the test set D_F, where the images were passed through the target model M_T to obtain the output vectors used for reconstruction. It is important to reiterate that although the output vectors are obtained from M_S, the gradient reconstruction process only utilizes the shadow CNN model, M_S. Please refer back to Sections 3.7.1 and **??** for the workflow. The reconstructions were done five times, and the average SSIM and the MSE are reported in Table 3. Based on the validation set scores, the best initialization method for image I is class average initialization.

Although M_S can imitate the behavior of M_T (refer to Section 4.2), the exact weights in the models may differ, which leads to the decrease in SSIM and an increase in MSE when doing the gradient reconstructions with M_S on the test set as the output vectors are obtained from M_T. To facilitate comparison with the reconstructions from the TCNN models, the SSIM and MSE scores from the test set D_F are also shown in Table 3. Additionally, Figure 12 shows the best gradient reconstructions per class on the test set.

	Shadow Validation Set		Test Set	
Initialization Method	SSIM Average	MSE Average	SSIM Average	MSE Average
Random Pixels	0.0536 ± 0.0081	0.1032 ± 0.0042	0.0417 ± 0.0065	0.1136 ± 0.0047
Specific Class Instance	0.2476 ± 0.0138	0.0687 ± 0.0078	0.1846 ± 0.0141	0.0778 ± 0.0084
Class Average	**0.3014** ± 0.0079	**0.0601** ± 0.0037	**0.2409** ± 0.0068	**0.0672** ± 0.0021

Table 3: Image Initialization Impact on Gradient Reconstructions

Figure 9: Target Reconstructions from D_F

Figure 10: Random Noise Initialization Reconstructions from D_F

Figure 11: Specific Class Instance Initialization Reconstructions from D_F

Figure 12: Average Class Initialization Reconstructions from D_F

5.2 Best Model Configuration

Different TCNNs, denoted as M_I, were tested to see the best input combination for reconstructions. Each TCNN is trained on 20 epochs. SSIM is utilized as the loss function for training. Since higher SSIM values are the goal, the objective is to minimize Loss $= 1 -$ SSIM. The baseline approach passes the output vector from the CNN into the TCNN. With the utilization of the shadow model, for each output vector (O) there is now a flattened gradient reconstruction $(\tilde{X})$ and activation's of the shadow models last linear layer (A^L) available. There are seven combinations of the three inputs. Each model was trained five times, and the average SSIM over the test set D_F was reported. Refer back to Section **??** to see how D_F flows through the workflow. In Table 4, it can be seen that passing just the gradient reconstruction $(\tilde{X})$ as input to the TCNN provides the best reconstructions with an average SSIM of 0.3727 and average MSE of 0.0649 on the test set.

Model	SSIM Average	MSE Average
$M_I(O)$	0.2298 $\pm$ 0.0041	0.0966 $\pm$ 0.0011
$M_I(A^L)$	0.1218 $\pm$ 0.0043	0.1077 $\pm$ 0.0014
$M_I(\tilde{X})$	**0.3727 $\pm$ 0.0030**	**0.0649 $\pm$ 0.0009**
$M_I(O + A^L)$	0.1619 $\pm$ 0.0027	0.1055 $\pm$ 0.0016
$M_I(O + \tilde{X})$	0.2467 $\pm$ 0.0013	0.0904 $\pm$ 0.0013
$M_I(A^L + \tilde{X})$	0.1822 $\pm$ 0.0051	0.0961 $\pm$ 0.0012
$M_I(O + A^L + \tilde{X})$	0.2194 $\pm$ 0.0016	0.0927 $\pm$ 0.0011

Table 4: SSIM & MSE Averages for Different Input Variations on Test Set D_F

The results disproved the belief that increasing the input features for a TCNN would lead to reconstructions with higher SSIM and lower MSE. A possible explanation for this unexpected outcome is the potential for increased input complexity to introduce additional noise and redundancy, thereby

hindering the model's ability to learn effectively. When combining multiple inputs, the TCNN may struggle to differentiate the most relevant features, leading to suboptimal performance. In contrast, the gradient reconstruction ($\tilde{X}$) alone likely offers a more concise and informative representation, enabling the TCNN to focus on the most pertinent details for accurate image reconstruction. This finding underscores the importance of carefully selecting input features to balance the richness of information with simplicity, ultimately facilitating more effective learning and better reconstruction quality. It was shown above that M_I, the inverse model of M_S performs the best when passing the

Figure 13: Target Reconstructions from D_F

Figure 14: $M_I(\tilde{X})$ Reconstructions from D_F

gradient reconstructions, $\tilde{X}$ as input to M_I. $M_I(\tilde{X})$ achieves an average SSIM value of 0.3727 and an average MSE value of 0.0649 on the test set, as shown in Table 4. The optimization approach, utilizing gradient-based optimization, achieved an average SSIM value of 0.2409 and an average MSE value of 0.0627 on D_F, as shown in Table 3. The combination of the gradient-based optimization with the inversion model ($M_I(\tilde{X})$) has a 54.73% increase in SSIM in comparison with the plain optimization approach ($\tilde{X}$). $M_I(\tilde{X})$ also has a 62.19% increase in SSIM in comparison to the standard inverse model ($M_I(O)$). The best reconstructions on D_F from $M_I(\tilde{X})$ per class are shown in Figure 14. They provide clearer reconstructions that also mimic more closely the luminance, contrast, and structure of the target reconstruction than the pure gradient reconstructions displayed in Figure 12. To see the best reconstructions on D_F from the other M_I models shown in Table 4, go see Appendix B.

5.3 Effect of Output Vector Rounding

Output vector rounding is one possible defense technique against the MI attack outlined in this work. This method provides the end user only access to the output vector, where each value is rounded to a given decimal point defined as d. To see the effect rounding has on the inverse model (TCNN) approach, d is set to 1. It is important to note that the gradient reconstructions are also constructed based on the rounded output vector.

Firstly, it was investigated how rounding the output vectors to one decimal place impacted a baseline TCNN model that only obtains the output vector, denoted as $M_I(O)$. Afterward, it was investigated what the impact is on the best input combination of passing the gradient reconstruction into the TCNN, denoted as $M_I(\tilde{X})$ (reference section 5.2). Table 5 shows that both models' average SSIM and MSE, after five iterations, stay the same regardless of rounding the output vector values.

There are multiple possible explanations for why the output vector rounding has no significant impact. TCNNs have a level of robustness that withstands small perturbations to the input data. The rounding effect adds a noise level but remains within the tolerance range; hence, the overall structure and

	Non-Rounded		Rounded	
Model	SSIM avg.	MSE avg.	SSIM avg.	MSE avg.
$M_I(O)$	0.2287 $\pm$ 0.0031	0.0977 $\pm$ 0.0012	0.2295 $\pm$ 0.0025	0.0965 $\pm$ 0.0015
$M_I(\tilde{X})$	0.3720 $\pm$ 0.0043	0.0659 $\pm$ 0.0005	0.3716 $\pm$ 0.0060	0.0658 $\pm$ 0.0013

Table 5: Output Vector Rounding: SSIM & MSE averages for $M_I(O)$ and $M_I(\tilde{X})$

patterns are preserved. One would think that the rounded output vectors would impact the gradient reconstructions ($\tilde{X}$) and hence the reconstructions obtained from $M_I(\tilde{X})$. However, the gradient values undergo only minor perturbations that do not significantly impact the model's ability enough to result in worse reconstructions.

5.4 Effect of Truncation

Another defense mechanism is truncating the output vector. From the output vector O, let k be the dimension of the partial prediction. O is then truncated to k dimensions to obtain O_k(i.e., preserving the top k scores and setting the rest to 0). The gradient reconstructions are obtained based on O_k.

To investigate the impact of truncation on the TCNN reconstructions, similar to the output vector rounding experiment, two models will be investigated. The first is the baseline TCNN, which uses only the output vector as input and is denoted as $M_I(O)$. The second model is the best TCNN input configuration, which uses gradient reconstruction as input and is denoted as $M_I(\tilde{X})$. As there are ten classes in the MNIST dataset, it was decided to compare the average SSIM and MSE values over the test set D_F where $k = 10, 5, 3, 1$. Each TCNN is trained five times and assumes k is known to the attacker.

	SSIM Average			
Model	k_{10}	k_5	k_3	k_1
$M_I(O)$	0.2538 $\pm$ 0.0025	0.3895 $\pm$ 0.0024	0.4092 $\pm$ 0.0018	0.4135 $\pm$ 0.0013
$M_I(\tilde{X})$	0.3747 $\pm$ 0.0015	0.4084 $\pm$ 0.0014	0.4172 $\pm$ 0.0027	0.4296 $\pm$ 0.0017
	MSE Average			
Model	k_{10}	k_5	k_3	k_1
$M_I(O)$	0.0952 $\pm$ 0.0003	0.0604 $\pm$ 0.0008	0.0580 $\pm$ 0.0007	0.0569 $\pm$ 0.0004
$M_I(\tilde{X})$	0.0649 $\pm$ 0.0006	0.0588 $\pm$ 0.0003	0.0572 $\pm$ 0.0005	0.0548 $\pm$ 0.0005

Table 6: Output Vector Truncation: SSIM & MSE averages for $M_I(O)$ and $M_I(\tilde{X})$

The results, presented in Table 6, reveal notable trends in the average SSIM and MSE values, providing insights into the impact of truncation.

For the baseline TCNN model, $M_I(O)$, the average SSIM values increase from 0.2538 to 0.3895 as the output vector is truncated from $k = 10$ to $k = 5$. This improvement continues slightly as k is further reduced to 3 and 1, ending at an average SSIM value of 0.4135. The increase in SSIM values with fewer k indicates that the reconstructions become more generalized, focusing on the primary characteristics of the specific class, which results in higher perceived structural similarity. The generalization of reconstructions can be seen in Figures 16, 17, 18, 19.

Figure 15: Target Reconstructions from D_F

Figure 16: $M_I(O)$: $k = 10$

Figure 17: $M_I(O)$: $k = 5$

Figure 18: $M_I(O)$: $k = 3$

Figure 19: $M_I(O)$: $k = 1$

The gradient reconstruction model, $M_I(\tilde{X})$, begins with a SSIM of 0.3747 at $k = 10$. The SSIM values continue to increase as k decreases and end at an SSIM value of 0.4296 at $k = 1$. While the SSIM values improve with fewer k, it is essential to note that this improvement is due to the generalization of the reconstructions. The images with higher k retain more fine details, which are essential for capturing the subtle features of the original images. As k decreases, these finer details are lost, resulting in more generalized but more structurally similar reconstructions. For instance, take a look at the number "2" in Figure 20. This particular "2" lacks the loop at the bottom, which is usually present in its standard form. When provided the full output vector ($k = 10$), the reconstruction captures the characteristics of the target image well, as shown in Figure 21. As k decreases, the loop at the bottom of the "2" is developed, shown in Figures 22, 23, 24. The loop at the bottom of the "2" is a main characteristic of the class average of the number "2" (reference Figure 4c). This is just one explanation for when k decreases, the reconstructions will converge to their class average representation, resulting in higher perceived structural similarity.

Figure 20: Target Reconstructions from D_F

Figure 21: $M_I(\tilde{X})$: k=10

Figure 22: $M_I(\tilde{X})$: k=5

Figure 23: $M_I(\tilde{X})$: k=3

Figure 24: $M_I(\tilde{X})$: k=1

To summarize, truncating the output vector is an effective defense mechanism against reconstructions. Although SSIM values increase as the dimensions of the output vector decrease, this comes at the cost of losing the finer details that make the images unique within their class. This loss of detail aligns with the overall goal of truncation: to generalize the reconstructions and reduce the specificity that makes one able to differentiate between different instances in the same class.

6 Conclusion

The concept of MI attacks was the focus of this research. MI attacks have branched into two directions. The first is the optimization approach, which inverts a model using gradient-based optimization in the data space. The second direction is the training of the second model, which acts as the inverse of the original model. This work combined both approaches and tested them in a non-white box framework, which entails no direct access to the target model parameters and weights. The main research question was: "How do different factors influence the fidelity of image reconstruction using transposed CNNs within a non-white box framework?"

The research incorporated shadow models, which played a critical role in the reconstruction process. These shadow models were essential in a gray-box setting, where the attacker's knowledge is limited to the target model's architecture but not its weights. Training a shadow model on a disjoint dataset from the same distribution as the target model's training data made it possible to mimic the target model's behavior. This approach allowed for effective reconstructions without access to the target model's internal parameters.

To address the main research question, the first sub-question (RQ1) explored how different input combinations to the TCNN—namely the output vector, gradient reconstruction, and activations from the last linear layer—affect reconstruction quality. The study demonstrated that incorporating gradient reconstructions improves the fidelity of image reconstructions. The TCNN model that used only the gradient reconstruction (denoted as $M_I(\tilde{X})$) achieved the highest SSIM average of 0.3727 ± 0.0030 and an MSE average of 0.0649 ± 0.0009 on the test set D_F.

The study also showed that combining the TCNN architecture with the gradient-based reconstructions proved superior to the standalone gradient-based method on the test set, with an average SSIM increase of 54.73%. This significant improvement underscores the TCNN's ability to preserve spatial details better than a gradient-based optimization approach. The improvement also highlights the benefit of combining both approaches of MI attacks outlined above.

To address the main research question further, the second sub-question (RQ2) examined how defensive mechanisms, such as truncation and output vector rounding, impact the quality of reconstructed images. Output vector rounding to one decimal place showed no significant deterioration in reconstruction quality, indicating the robustness of the TCNN model to minor perturbations in the input. Truncation of the output vector to the top-k scores (with $k = 10, 5, 3, 1$) revealed that while SSIM values increased with fewer dimensions, this came at the cost of losing finer details in the reconstructions, leading to more generalized images.

It is essential to acknowledge the limitations of this study. The reliance on the MNIST dataset [15], while necessary for the initial exploration, does limit the generalizability of the findings to more complex data. Additionally, the study primarily focused on relatively simple model architectures, which may only partially represent the diversity of real-world models. More complex models, such as deeper networks or those with different structures (e.g., ResNet, EfficientNet), might respond differently to the input combinations and defensive mechanisms tested. Another limitation of the work is the assumption that the attacker has access to a dataset from the same distribution as the target dataset, as this assumption may not always hold in a real-world setting. These considerations are crucial for comprehensively understanding the research's scope and implications.

7 Future Work

Looking ahead, the potential for further exploration and advancement is vast. Future work should evaluate the proposed methods on more complex datasets, such as CIFAR-10 [12] or ImageNet [25], to understand the scalability and effectiveness across different types of data. Doing so would provide insight into whether the proposed MI attack is applicable in more diverse contexts. This work attempted to extract more information from the shadow model to aid the reconstruction process of the inversion model (the TCNN model). The last linear activations were calculated for each output vector based on the trained and frozen weights and bias from the shadow model. Although inputting these activations into the TCNN yielded poor reconstructions, the concept of extracting information from the shadow model should be further investigated. One possible option is to extract the shadow model's last feature map based on a given output vector and pass it to the inversion model to see if it yields better reconstructions than those obtained from the last linear layer activations.

Moreover, while truncation and rounding were somewhat effective, it is vital to continue testing further defense mechanisms against MI attacks. One potential defense mechanism that could distort reconstructions is adding a level of noise to the output vector. Another potential defense mechanism could be training a secondary (student) model to approximate the original (teacher) model's outputs but with reduced precision or information leakage. Reconstructions on the student model may still be possible, but they lack privacy-invasive details that may be apparent in the reconstructions based on the teacher model.

The fusion of the two branches of MI attacks in this work has shown the possibility of recognizable reconstructions. This work has opened the door for the future work described above. This work aims to bring awareness to these types of MI attacks and the countermeasures one can implement to prevent the potential of privacy-invasive reconstructions. Although the current state of reconstructions do not pose an imminent threat, the topic of MI attacks continues to grow. Therefore, monitoring this issue and implementing suggested countermeasures to protect data privacy is crucial.

References

[1] Michael Affenzeller et al. "White Box vs. Black Box Modeling: On the Performance of Deep Learning, Random Forests, and Symbolic Regression in Solving Regression Problems". In: *Computer Aided Systems Theory – EUROCAST 2019*. Ed. by Roberto Moreno-Díaz, Franz Pichler, and Alexis Quesada-Arencibia. Cham: Springer International Publishing, 2020, pp. 288–295. ISBN: 978-3-030-45093-9.

[2] Shams Ahmed et al. "Deep learning modelling techniques: current progress, applications, advantages, and challenges". In: *Artificial Intelligence Review* 56 (Apr. 2023). DOI: 10.1007/s10462-023-10466-8.

[3] Rishi Bommasani et al. *The Foundation Model Transparency Index*. 2023. arXiv: 2310.12941 [cs.LG].

[4] Bostjan Brumen et al. "Outsourcing Medical Data Analyses: Can Technology Overcome Legal, Privacy, and Confidentiality Issues?" In: *J Med Internet Res* 15.12 (Dec. 2013), e283. ISSN: 14388871. DOI: 10.2196/jmir.2471. URL: http://www.ncbi.nlm.nih.gov/pubmed/24342053.

[5] Shan Chang and Chao Li. "Privacy in Neural Network Learning: Threats and Countermeasures". In: *IEEE Network* 32.4 (2018), pp. 61–67. DOI: 10.1109/MNET.2018.1700447.

[6] Chao Dong et al. "Image Super-Resolution Using Deep Convolutional Networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.2 (2016), pp. 295–307. DOI: 10.1109/TPAMI.2015.2439281.

[7] Alexey Dosovitskiy and Thomas Brox. *Inverting Visual Representations with Convolutional Networks*. 2016. arXiv: 1506.02753 [cs.NE].

[8] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021. arXiv: 2010.11929 [cs.CV].

[9] Ehsan Fathi and Babak Maleki Shoja. "Chapter 9 - Deep Neural Networks for Natural Language Processing". In: *Computational Analysis and Understanding of Natural Languages: Principles, Methods and Applications*. Ed. by Venkat N. Gudivada and C.R. Rao. Vol. 38. Handbook of Statistics. Elsevier, 2018, pp. 229–316. DOI: https://doi.org/10.1016/bs.host.2018.07.006. URL: https://www.sciencedirect.com/science/article/pii/S016971611830021X.

[10] 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*. CCS '15. Denver, Colorado, USA: Association for Computing Machinery, 2015, pp. 1322–1333. ISBN: 9781450338325. DOI: 10.1145/2810103.2813677. URL: https://doi.org/10.1145/2810103.2813677.

[11] C.A. Jensen et al. "Inversion of feedforward neural networks: algorithms and applications". In: *Proceedings of the IEEE* 87.9 (1999), pp. 1536–1549. DOI: 10.1109/5.784232.

[12] Alex Krizhevsky. *Learning Multiple Layers of Features from Tiny Images*. Tech. rep. University of Toronto, 2009. URL: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

[13] "L2 Regularization Model with Removal of Gaussian Noise". In: *Journal of mathematics and informatics* 23 (2022), pp. 41–50. DOI: 10.22457/jmi.v23a04211.

[14] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *Nature* 521 (2015), pp. 436–444. DOI: 10.1038/nature14539. URL: https://doi.org/10.1038/nature14539.

[15] Yann LeCun, Corinna Cortes, and CJ Burges. "MNIST handwritten digit database". In: *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist* 2 (2010).

[16] Geert Litjens et al. "A survey on deep learning in medical image analysis". In: *Medical Image Analysis* 42 (2017), pp. 60–88. ISSN: 1361-8415. DOI: https://doi.org/10.1016/j.media.2017.07.005. URL: https://www.sciencedirect.com/science/article/pii/S1361841517301135.

[17] Jun Ma et al. "Loss odyssey in medical image segmentation". In: *Medical Image Analysis* 71 (2021), p. 102035. ISSN: 1361-8415. DOI: https://doi.org/10.1016/j.media.2021.102035. URL: https://www.sciencedirect.com/science/article/pii/S1361841521000815.

[18] Aravindh Mahendran and Andrea Vedaldi. "Understanding deep image representations by inverting them". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 5188–5196. DOI: 10.1109/CVPR.2015.7299155.

[19] Kenta Moriwaki et al. *Hybrid Loss for Learning Single-Image-based HDR Reconstruction*. 2018. arXiv: 1812.07134 [cs.CV].

[20] Charlie Nash, Nate Kushman, and Christopher K. I. Williams. *Inverting Supervised Representations with Autoregressive Neural Density Models*. 2019. arXiv: 1806.00400 [stat.ML].

[21] Nicolas Papernot et al. *Towards the Science of Security and Privacy in Machine Learning*. 2016. arXiv: 1611.03814 [cs.CR].

[22] Joaquin Quiñonero-Candela et al. *Dataset Shift in Machine Learning*. Neural Information Processing. Cambridge, Massachusetts and London, England: MIT Press, 2009. ISBN: 978-0-262-17005-5. URL: http://www.acad.bg/ebook/ml/The.MIT.Press.Dataset.Shift.in.Machine.Learning.Feb.2009.eBook-DDU.pdf.

[23] Andrew J. Reader et al. "Deep Learning for PET Image Reconstruction". In: *IEEE Transactions on Radiation and Plasma Medical Sciences* 5.1 (2021), pp. 1–25. DOI: 10.1109/TRPMS.2020.3014786.

[24] Leonid I. Rudin, Stanley Osher, and Emad Fatemi. "Nonlinear total variation based noise removal algorithms". In: *Physica D: Nonlinear Phenomena* 60.1 (1992), pp. 259–268. ISSN: 0167-2789. DOI: https://doi.org/10.1016/0167-2789(92)90242-F. URL: https://www.sciencedirect.com/science/article/pii/016727899290242F.

[25] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[26] Umme Sara, Morium Akter, and Mohammad Shorif Uddin. "Image Quality Assessment through FSIM, SSIM, MSE and PSNR—A Comparative Study". In: *Journal of Computer and Communications* 7.3 (2019), pp. 8–18. DOI: 10.4236/jcc.2019.73002. URL: https://www.scirp.org/pdf/JCC_2019030117485323.pdf.

[27] Ramprasaath R. Selvaraju et al. "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization". In: *International Journal of Computer Vision* 128.2 (Oct. 2019), pp. 336–359. ISSN: 1573-1405. DOI: 10.1007/s11263-019-01228-7. URL: http://dx.doi.org/10.1007/s11263-019-01228-7.

[28] Reza Shokri et al. *Membership Inference Attacks against Machine Learning Models*. 2017. arXiv: 1610.05820 [cs.CR].

[29] Chandra Thapa and Seyit Camtepe. "Precision health data: Requirements, challenges and existing techniques for data security and privacy". In: *Computers in Biology and Medicine* 129 (2021), p. 104130. DOI: 10.1016/j.compbiomed.2020.104130. URL: https://doi.org/10.1016/j.compbiomed.2020.104130.

[30] Annamária R. Várkonyi-Kóczy. "Observer-Based Iterative Fuzzy and Neural Network Model Inversion for Measurement and Control Applications". In: *Towards Intelligent Engineering and Information Technology*. Ed. by Imre J. Rudas, János Fodor, and Janusz Kacprzyk. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 681–702. ISBN: 978-3-642-03737-5. DOI: 10.1007/978-3-642-03737-5_49. URL: https://doi.org/10.1007/978-3-642-03737-5_49.

[31] Zhou Wang et al. "Image quality assessment: from error visibility to structural similarity". In: *IEEE Transactions on Image Processing* 13.4 (2004), pp. 600–612. DOI: 10.1109/TIP.2003.819861.

[32] L. de-Wit and J. Wagemans. "Visual Perception". In: *Encyclopedia of Human Behavior: Second Edition* (Jan. 2012), pp. 665–671. DOI: 10.1016/B978-0-12-375000-6.00371-2.

[33] Lingfei Wu et al. "Graph Neural Networks for Natural Language Processing: A Survey". In: *Foundations and Trends® in Machine Learning* 16.2 (2023), pp. 119–328. ISSN: 1935-8237. DOI: 10.1561/2200000096. URL: http://dx.doi.org/10.1561/2200000096.

[34] Stephen Wu et al. "Deep learning in clinical natural language processing: a methodical review". In: *Journal of the American Medical Informatics Association* 27.3 (Dec. 2019), pp. 457–470. ISSN: 1527-974X. DOI: 10.1093/jamia/ocz200. eprint: https://academic.oup.com/jamia/article-pdf/27/3/457/34152802/ocz200.pdf. URL: https://doi.org/10.1093/jamia/ocz200.

[35] Ziqi Yang et al. "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. London, United Kingdom: Association for Computing Machinery, 2019, pp. 225–240. ISBN: 9781450367479. DOI: 10.1145/3319535.3354261. URL: https://doi.org/10.1145/3319535.3354261.

[36] Afia Zafar et al. "A Comparison of Pooling Methods for Convolutional Neural Networks". In: *Applied Sciences* 12.17 (2022). ISSN: 2076-3417. DOI: 10.3390/app12178643. URL: https://www.mdpi.com/2076-3417/12/17/8643.

[37] Yuheng Zhang et al. *The Secret Revealer: Generative Model-Inversion Attacks Against Deep Neural Networks*. 2020. arXiv: 1911.07135 [cs.LG].

[38] Xuejun Zhao et al. "Exploiting Explanations for Model Inversion Attacks". In: *Proceedings of the IEEE International Conference on Computer Vision* (Apr. 2021), pp. 662–672. ISSN: 15505499. DOI: 10.1109/ICCV48922.2021.00072. URL: https://arxiv.org/abs/2104.12669v3.

A Model Architectures

Both M_T and M_S utilize cross-entropy loss defined in Formula 16. The Adam optimizer with a learning rate of 0.001 is used to train the models. Their architectures are found in Figures 25a and 25b, respectively.

$$L_{\text{CE}}(I, y) = -\sum_{c=1}^{C} y_c \log(\text{softmax}(M(I))_c) \tag{16}$$

Within Formula 16, C represents the total number of classes in the classification problem (MNIST: $C = 10$). Here, y_c is either 0 or 1, indicating whether the class label is the correct classification. $M_(I)$ represents passing image I through the model M.

```
CNN(
  (layers): ModuleList(
    (0): Conv2d(1, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (3): ReLU()
    (4): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (5): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (6): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (7): ReLU()
    (8): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (9): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (10): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (11): ReLU()
    (12): Flatten(start_dim=1, end_dim=-1)
    (13): Linear(in_features=8192, out_features=50, bias=True)
    (14): Dropout(p=0.5, inplace=False)
    (15): Linear(in_features=50, out_features=10, bias=True)
  )
)
```

(a) M_T Architecture

```
CNN(
  (layers): ModuleList(
    (0): Conv2d(1, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (3): ReLU()
    (4): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (5): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (6): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (7): ReLU()
    (8): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (9): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (10): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (11): ReLU()
    (12): Flatten(start_dim=1, end_dim=-1)
    (13): Linear(in_features=8192, out_features=50, bias=True)
    (14): Dropout(p=0.5, inplace=False)
    (15): Linear(in_features=50, out_features=10, bias=True)
  )
)
```

(b) M_S Architecture

Figure 25: CNN Architectures

M_I utilizes SSIM as the loss function during training. Since higher SSIM values are the goal, the objective is to minimize Loss $= 1 -$ SSIM. The optimizer of choice was Adam, with a learning rate of 0.001. The architecture is found in Figure 26.

```
TCNN(
  (layers): ModuleList(
    (0): ConvTranspose2d(10, 512, kernel_size=(4, 4), stride=(1, 1))
    (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): Tanh()
    (3): Dropout(p=0.3, inplace=False)
    (4): ConvTranspose2d(512, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (5): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (6): Tanh()
    (7): Dropout(p=0.5, inplace=False)
    (8): ConvTranspose2d(256, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (9): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (10): Tanh()
    (11): Dropout(p=0.5, inplace=False)
    (12): ConvTranspose2d(128, 1, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (13): Sigmoid()
  )
)
```

Figure 26: TCNN Architecture, the input channel for the first layer changes depending on the input provided. $M_I(O)$: channels=10. $M_I(A^L)$: channels=50, $M_I(\tilde{X})$: channels=1024

B Auxiliary Images

Figure 27: $M_I(O)$ Reconstructions (row 1: target images, row 2: reconstructions)

Figure 28: $M_I(A^L)$ Reconstructions (row 1: target images, row 2: reconstructions)

Figure 29: $M_I(O + A^L)$ Reconstructions (row 1: target images, row 2: reconstructions)

Figure 30: $M_I(O + \tilde{X})$ Reconstructions (row 1: target images, row 2: reconstructions)

Figure 31: $M_I(A^L + \tilde{X})$ Reconstructions (row 1: target images, row 2: reconstructions)

Figure 32: $M_I(O + A^L + \tilde{X})$ Reconstructions (row 1: target images, row 2: reconstructions)