Defending Model Inversion Attack Using an Improved Filter-Based Approach

Ananta Raha[1], Junaeid Ahmed[1], Md Shohrab Hossain[1], and Suryadipta Majumdar[2]

[1] Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Bangladesh
{ananta.raha.99, junaeidahmed1303}@gmail.com, shohrab@cse.buet.ac.bd
[2] Concordia Institute for Information Systems Engineering, Concordia University, Canada
suryadipta.majumdar@concordia.ca

Abstract. In a model inversion attack, an adversary tries to reconstruct private training data through iterative inference of a neural network model. To ensure confidentiality of the training data, protection against model inversion attacks is crucial. However, existing defense techniques primarily require modifications to the trained model architecture or even retraining, which limits their applicability to deployed models. In addition, most of them become ineffective against label-only attacks, which require minimal information to succeed. This paper proposes an improved defense mechanism that filters out inference requests from malicious attackers. It does not alter existing model architectures and works against various types of attacks. Our experimental results show that our approach is highly effective with mean defense accuracy scores of 90.00% and 95.83% on two datasets, respectively. Therefore, the proposed approach has been proven to be successful in defending against contemporary model inversion attacks, thus achieving the objectives of this study.

Keywords: Model inversion attack · ML privacy · Label-only attack · Server-side defense · ML attacks · Input reconstruction

1 Introduction

With the advancement of artificial intelligence (AI) and machine learning (ML), deep neural networks have become a vital instrument for numerous applications, including speech recognition, image classification, and natural language processing, often achieving human-level performance. [6, 15]. Nevertheless, models are prone to various privacy attacks, such as membership inference [11], model stealing [19], and model inversion [3]. Due to its intrinsic characteristics, the model inversion (MI) attack is the most difficult of all privacy threats against ML models [13]. Using this attack, the attacker can deduce sensitive training data with minimal information [4]. Therefore, it is crucial to study recent trends in MI attacks and responses along with potential future directions [2].

Existing defense techniques (e.g., [5, 21, 25]) against MI attacks require altering the model architecture or filtering the confidence vector. However, they become ineffective against label-only attacks [26], a variant of the MI attack that requires only labels. Since labels are public outputs produced through model inference, these attacks have become difficult to prevent.

Therefore, we aim to develop a robust defense architecture against MI attacks, including its label-only variants. This paper proposes a straightforward defense approach that evaluates the similarity between consecutive input submissions and effectively controls model inference based on that outcome. Malicious inferences serve as the backbone of any MI attack. By reliably filtering them, our approach prevents potential data leakage in model inversion attacks, while keeping the model intact. For simplicity, we employed the MI attack by Fredikson [3] on the MNIST [9] and CelebA [14] datasets to validate our proposed system. Our results show that the proposed technique is highly effective against MI attacks yet flexible enough to permit regular user inferences, yielding good precision scores. That demonstrates the robustness of our suggested defense technique.

In short, the contributions in this paper are as follows:

- We develop a filter-based MI defense mechanism that does not require model modification or re-training. To the best of our knowledge, this is the first attempt to apply a similarity-based filtering method for avoiding model re-training and alteration on server.
- We demonstrate that our proposed defense mechanism is effective against both label-only and regular MI attacks. This universality is achieved automatically by relying on the communication channel rather than model output or architecture.

The remainder of the paper is organized as follows. Section 2 highlights the gaps in the existing defense methods. In Section 3, our proposed defense mechanism is explained. Sections 4 and 5 contain the specifics of our experiments and results, respectively. Finally, a highlight of our work and potential future directions are discussed in Section 6.

2 Background & Literature Review

This section explores various MI attacks, existing defense approaches, and the research gaps present in mitigating these attacks.

2.1 Terminologies

Model Training & Inference Model training, a necessary part of the ML lifecycle, involves learning patterns in the training data supplied. Inference means using a trained model to generate predicted confidence scores or labels.

Model Inversion Attack During training, NN models may unintentionally recall information regarding training data [1]. In an MI attack, the attacker aims to reconstruct sensitive training data through iterative inferences [4]. For example, an adversary can recreate individual or generic pictures of a class in case of image data, or infer sensitive properties from training samples for tabular training data.

White-Box and Black-Box Attacks An attack can have two configurations: white-box or black-box [17]. In a black-box attack, the adversary can only query the target model without knowing the specifics, such as model weights and gradients [18, 22]. A white-box attack allows adversaries to execute attacks more effectively since they are fully aware of the model architecture and specific parameters [2, 10, 16].

Reconstruction and Inference Cycle In general, MI attacks have two phases: reconstruction and inference. Starting with an attacker-initialized image, consecutive repeats of the stages eventually produce samples identical to the training data [23].

2.2 Existing Works

Foundation of MI Attacks Fredrikson et al. [4] introduced MI attacks where an adversary could infer sensitive genetic information from a linear regression model. Their further work on image data [3] revealed that attackers could exploit confidence scores from model outputs to recreate sensitive training data, such as face images from facial recognition systems. These were all white-box attacks since the adversary possessed the complete information of the model architectures.

Contemporary MI Attacks The work of Zhang et al. [24] discussed various works on MI attack defense and privacy threats against cloud-based neural networks. They referred to two MI attack strategies: learning-based, which inverts a classifier model by creating an inverse architecture, and optimization-based, which calculates optimal information regarding the target class [24]. A notable work by Zhu et al. [26] introduced an unavoidable flaw in Neural Network (NN) models in black-box and white-box scenarios. They simulated MI attacks with the least amount of knowledge (only labels) and achieved outcomes close to the original training data. Fig. 1 illustrates some of them. Unfortunately, there is no efficient defense against this black-box attack. The vulnerability of Large Vision-Language Models (LVLM) has been discussed in the study of Liu et al. [12], where the authors admitted that continuous activity monitoring is required to restrict attacker's query and exploitation.

Fig. 1. Attack outputs from literature [26]

Current Defense Strategies Jegorova et al. [8] presented a comprehensive on data leakage, current techniques, challenges, and unresolved issues in the field of data security. A thorough guide to improve Federated Learning (FL) security against privacy attacks was provided in [7]. They demonstrated new techniques for quantifying and visualizing possible data leakage from FL [7]. Gong et al. [5] introduced NetGuard, a novel defense module against MI attacks that protects model by injecting exclusively-created fake samples during training. However, this approach requires the model to be re-trained. Yang et al. [21] suggested a technique that purifies the confidence vector of a classifier to reduce dispersion, making it difficult for attackers to deduce confidential training data. Zhou et al. [25] proposed a defense strategy based on data poisoning, which requires altering the final output vector of the model. In the extensive study of [20], they review various MI defense methods, such as feature obfuscation, gradient pruning and obfuscation, and model enhancement. However, all these require model alteration to a certain extent.

Overall Gap Analysis Existing literature contains the following gaps:

- Most of the defense approaches require re-training of the models which is infeasible in most cases.
- Some defense approaches poisons output vector, yet remaining useless for label-only attacks.
- Labels are the least information generated by models and cannot be altered, which makes these attacks almost inevitable.
- No effective defense strategy against label-only MI attacks was found in our extensive study.

This paper addresses these issues by incorporating a balanced defense module that can be integrated easily with server-side models. Since heavy filtering and buffering require significant resources on the server, our approach offers flexibility, incurring negligible server overhead.

3 Defense Mechanism

Subsequent sections elaborate the complete defense approach and its applicability on different attack types.

3.1 Overview

Existing defense approaches are ineffective against label-only attacks, since all models inherently expose output labels. Once returned to the client, these labels can be exploited with minimal effort. While attackers cannot tamper with the inference pipeline itself, they can repeatedly query the model to obtain labels and initiate the attack effortlessly. To counter this, we introduce a lightweight pre-inference filter that evaluates the similarity of successive inputs and halts processing when suspiciously repetitive queries are detected (Fig. 2). This mechanism prevents systematic probing without modifying the core model architecture. At the same time, it is designed to minimize false detections and ensure legitimate users are not penalized. The resulting framework preserves model utility while enabling flexible tuning against different model inversion (MI) attack strategies.

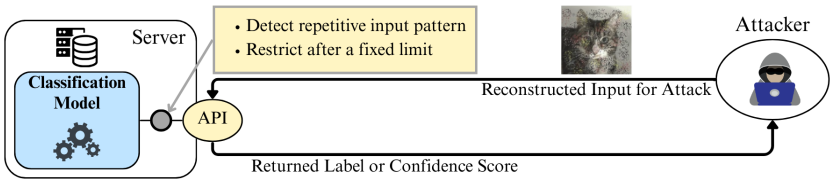

Fig. 2. Our key idea of detecting attacker based on their input pattern

In our particular case of image data, attackers and end-users transmit input images to the server over the network, as illustrated in Fig. 3. The submitted image initially undergoes the defense filter in the server, which marks it as either safe or restricted. Safe images go through model inference, resulting in regular server response. In case of restriction, the server responds with a rejection message. Algorithm 1 gives a logical view of this process. The uploaded input image,

Algorithm 1 Client Session Handling

Initialize buffer $B \leftarrow \emptyset$
Initialize counter $\texttt{count} \leftarrow 0$
while True **do**
  $x_{input} \leftarrow$ Fetch image from client
  $(\texttt{flag}, \texttt{count}) \leftarrow \texttt{Filter}(x_{input}, B, \texttt{count})$
  if $\texttt{flag} = \texttt{Flag(UNSAFE)}$ **then**
    Mark client as attacker
    Terminate session
    break
  end if
  $y_{label} \leftarrow \texttt{Model}(x_{input})$
  Send y_{label} back to client
end while

x_{input} is passed to the `Filter` method, which returns either `SAFE` or `UNSAFE` flag. In the case of `UNSAFE`, the server terminates the session, marking it as an attack. Otherwise, x_{input} is fed to the model, and predicted label y_{label} is supplied with regular server response. Subsequent paragraphs provide a detailed explanation of the proposed filter, similarity metrics, and applicability of the proposed method.

3.2 Similarity Metrics

Our proposed defense approach determines the similarity of two images using two popular measures, SSIM (Structural Similarity Index Measure) and Simple Hash. SSIM evaluates brightness, contrast, and structural similarity between two images, yielding a result between -1 and 1 inclusive. The value of 1 indicates that the images are identical, whereas -1 signifies the highest degree of dissimilarity. Simple Hash calculates the average pixel intensity after downsampling the image. SSIM is appropriate because we aim to detect the minor pixel-level alterations made by attackers when developing their attack models and infrastructure. Besides, we considered mean intensity values to minimize false positives.

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{1}$$

Inside the defense filter, two parameters were defined: `ssim_threshold` for SSIM and `hash_threshold` for Simple hash. The similarity between two photos is determined by the hash difference and the SSIM value. If both values exceed the thresholds, the images are deemed similar. SSIM is given by Eqn. (1). The symbols μ_x and μ_y represent the average intensities of images x and y, respectively. The variances are denoted as σ_x^2 and σ_y^2, while σ_{xy} indicates the covariance between the two signals. Constants C_1 and C_2 are employed to prevent division by zero, defined as $C_1 = (k_1 L)^2$ and $C_2 = (k_2 L)^2$. L denotes the dynamic range of pixel intensities (e.g., 255 in 8-bit images), while k_1 and k_2 are small constants utilized for numerical stability.

3.3 Proposed Filter Mechanism

Fig. 3 contains the flowchart of the proposed filter mechanism. The filter maintains a buffer and a count of suspicious requests specific to each client session. At first, it verifies that the client was not already blocked previously by checking the client-specific count against the allowed limit. Afterward, it checks for similarity against all images stored in the buffer using metrics described in Section 3.2. Dissimilar images are queued in the buffer and sent to the model for inference and additional processing. Conversely, a similar image increases the suspicious count before enqueuing into the buffer. Once it exceeds the limit, the client becomes forbidden. Looking at the underlying Algorithm 2 gives a more precise logical view of the process above. At first, we check suspicious `count` against the allowed limit L_{sp}. Then SSIM and Hash values between the input image x_{input} and every image stored in buffer B are compared to the thresholds

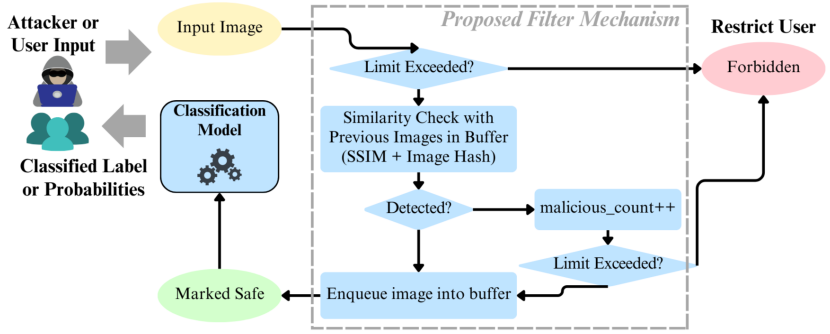

Fig. 3. Communication between attacker and server model with proposed defense mechanism

Algorithm 2 Filter

Require: Input sample x_{input}, buffer B, malicious counter `count`, thresholds T_{ssim}, T_{hash}, limit L_{sp}

if `count` $\geq L_{sp}$ **then**
  return `Flag(UNSAFE), count`
end if
for each image x_i in buffer B **do**
  if $\text{SSIM}(x_i, x_{input}) > T_{ssim}$ **and** $|\text{hash}(x_i) - \text{hash}(x_{input})| < T_{hash}$ **then**
    `count` $\leftarrow$ `count` $+ 1$
    if `count` $\geq L_{sp}$ **then**
      return `Flag(UNSAFE), count`
    end if
  end if
end for
`enqueue`(B, x_{input})
return `Flag(SAFE), count`

T_{ssim} and T_{hash}. Once it exceeds both thresholds, `count` is increased by 1. If `count` exceeds L_{sp}, the flag `UNSAFE` is returned. Otherwise, it returns the `SAFE` flag after queuing x_{input} into B.

3.4 Applicability For Different Attack Types

All MIA attacks illustrated in Fig. 4 follow a similar pattern. Alongside being repetitive, there are high similarities between consequent inputs submitted by an attacker. Hence, our proposed filter applies to all types of attacks, whether label-only or not. Also, our defense mechanism is appropriate for any model architecture since the filter does not require modifying the confidence vector or gradient. It resides before the model inference step allowing us to integrate it without any structural modification to the original NN model.

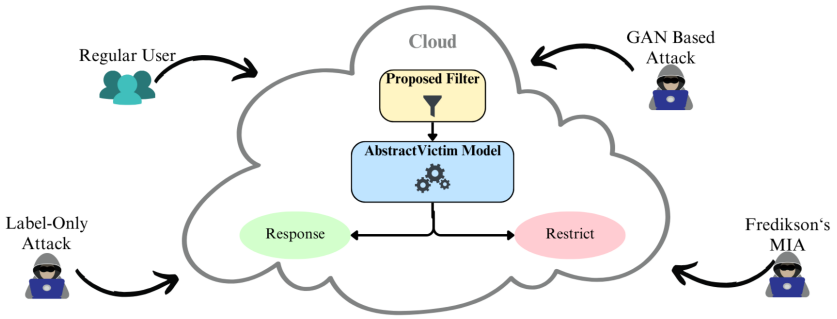

Fig. 4. Applicability of proposed defense for various attack types

4 Experiment

Our experiments are based on the simulation of attackers and regular users. In both cases, the model has been equipped with our proposed defense filter. We explain the details of our experimental setup in the following subsections.

4.1 Prior Assumptions

Attackers and regular users have no direct access to the neural network model inference process or the defense mechanism. The model and proposed defense mechanism reside on a remote server, and the inference process remains uncompromised. Furthermore, this defense technique applies to both black-box and white-box attacks, as both involve repeatedly querying the server model 4.

4.2 Selection of Attack

Our defense approach is independent of the attack types (Section 3.4). Therefore, we select the most straightforward model inversion attack by re-implementing the work of Fredrikson et al. [3]. We keep the confidence matrix exposed to align with this attack strategy; nevertheless, it has no impact on evaluating the operation of our defense system. We have used LENet5 [9] and a standard CNN as the classification models (victim) for the two datasets, respectively.

4.3 Selection of Image Dataset

Two widely recognized datasets are utilized: MNIST (Modified National Institute of Standards and Technology) [9] and CelebA [14]. The MNIST dataset serves as a standard benchmark for the classification of handwritten digits. It comprises 70,000 grayscale images of ten digits (0–9), each measuring 28×28 pixels. We selected it to align with the original Fredikson's attack [3]. The CelebA dataset contains 202,599 images of 10,177 celebrities, each annotated with 40 binary attributes and 5 facial landmarks. It is widely used for face recognition, attribute

prediction, and generative modeling. To align our problem, we divide it into 4 distinct classes, by combining several features. We choose MNIST and CelebA to reproduce easily-attainable attacks, ensuring a fair evaluation of our defense.

4.4 Experimental Configurations

All the experiments and simulations are carried out in Jupyter notebooks on the Kaggle platform, utilizing a server PC with NVIDIA P100 GPU with 16 Gigabytes of RAM. Alongside this, standard Python libraries like TensorFlow, and OpenCV are used heavily for implementation of the experiments.

4.5 Simulation of Attacks & Regular User Inputs

In our experiment, an attack trial means unlimited iterations where each iteration involves the submission of an image sample to the model server, which houses the defense mechanism and the model inference pipeline. Upon successful inference, the confidence vector is supplied to the attacker. After optimizing the reconstructed image with confidence scores, the attacker stages it for the next iteration. Random noise is introduced before every iteration to complicate the server's ability to identify identical consecutive inputs. The trial continues until it gets restricted by our proposed defense system. For each label, we recorded the number of maximum iterations before restriction by the defense.

On the other hand, each regular input trial contains one random test image from the dataset for the particular label. In this case, we reduced the datasets to exactly 800 images per label to minimize class imbalance. It is worth mentioning that we did not apply noise to the submitted images to mimic average user submissions.

5 Results

The following portions include a description of the symbols and configurations used in our experiments, our evaluation criteria, detailed results of the simulations, and finally, an overall performance analysis of the proposed defense approach.

5.1 Notations

The following notations are used in this section.

TP True Positives, correctly detected attacks.
TN True Negatives, correctly identified non-attacks (regular user inputs).
FP False Positives, non-attacks incorrectly flagged as attacks.
FN False Negatives, attacks missed.
N_B Size of the buffer in filter mechanism
T_{ssim} Threshold for SSIM score in filter mechanism
T_{hash} Threshold for Simple Hash difference in filter mechanism
L_{sp} Allowed limit of suspicious submissions before restriction.

5.2 Performance Metrics

In order to assess the efficacy of our defense approach, we used accuracy, precision, recall, and F1-score in the experiments. Accuracy denotes the proportion of all inputs that are correctly identified. Precision indicates the proportion of the attack predictions (restricted inputs) that are actually attacks. Recall represents the percentage of attacks detected among actual attacks. F1-score provides a balanced evaluation of detection performance. Precision, recall, and accuracy demonstrate the effectiveness of the defense in identifying attacks. The F1-score perfectly reflects its permissive nature towards regular users.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4}$$

$$\text{F1-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5}$$

Table 1. Description of defense configurations in experiments

Parameter	**Configuration 1**	**Configuration 2**	**Configuration 3**
N_B	10	10	10
T_{ssim}	0.99990	0.99999	0.99900
T_{hash}	0.00010	0.00001	0.00100
L_{sp}(MNIST)	4	4	4
L_{sp}(CelebA)	8	8	8

5.3 Outcomes of Attack Simulation

For introducing randomness, three different configurations of the parameters N_B, T_{ssim}, T_{hash} and L_{sp} were used (Refer to Table 1). We set the parameter values based on a trial-and-error approach. In every configuration, multiple distinct attack trials were attempted: 30 in MNIST (three trials for each of the 10 labels) and 12 in CelebA (three trials for each of the four labels). Table 2 and 3 presents the dataset-specific results for individual configurations. For both datasets, the trials get caught by our defense technique at a point. Besides, configuration 1 and configuration 2 allowed large input submissions before detection because the higher T_{ssim} and lower T_{hash} values allowed only submissions with high similarity. In contrast, configuration 3 performed noticeably better with fast identification of attack. Because the values of T_{ssim} and T_{hash} were appropriate in this configuration to maintain a good balance between attack detection and false positives.

Table 2. Summary of Attack Attempts for the MNIST Dataset

												Total
Labels		'0'	'1'	'2'	'3'	'4'	'5'	'6'	'7'	'8'	'9'	10
Attack Attempts		3	3	3	3	3	3	3	3	3	3	30
Successful		0	0	0	0	0	0	0	0	0	0	0
Max Iter. Before Detection	**Config. 1**	390	845	1324	652	1015	1142	895	727	1115	1162	
	Config. 2	1275	1178	584	936	961	878	1060	911	1235	1005	
	Config. 3	68	308	52	57	83	43	71	133	80	82	

Table 3. Summary of Attack Attempts for the CelebA Dataset

						Total
Labels		'Male'	'Female'	'Bangs'	'Eyeglasses'	4
Attack Attempts		3	3	3	3	12
Successful		0	0	0	0	0
Max Iter. Before Detection	**Config. 1**	1478	1462	1127	1698	
	Config. 2	1581	2552	1733	1561	
	Config. 3	305	124	948	1489	

5.4 Outcomes of Regular User Simulation

Similar to the attack simulation (Section 5.3), we carried out regular user input trials for the three defense configurations: 30 in MNIST (three per label) and 12 in CelebA (three per label). The label-wise outcomes are illustrated in Table 4 and 5. The total number of false attack detections is noticeably the same in all configurations due to the balanced dataset explained in Section 4.5. For MNIST, complex digits like 8 and 9 confused the defense architecture since their delicate character appearance makes the similarity evaluation harder than the other labels. Similarly, in CelebA dataset, complex image features like bangs and eyeglasses makes the challenge harder.

Table 4. Summary of Regular User Attempts for MNIST Dataset

												Total
Labels		'0'	'1'	'2'	'3'	'4'	'5'	'6'	'7'	'8'	'9'	10
Regular User Attempts		3	3	3	3	3	3	3	3	3	3	30
False Detection	**Config. 1**	0	0	0	2	0	0	2	0	1	1	6
	Config. 2	1	0	1	0	1	0	0	0	1	1	5
	Config. 3	1	0	0	0	0	2	1	1	1	1	7

Table 5. Summary of Regular User Attempts for CelebA Dataset

						Total
Labels		'Male'	'Female'	'Bangs'	'Eyeglasses'	4
Regular User Attempts		3	3	3	3	12
False Detection	**Config. 1**	0	1	0	0	1
	Config. 2	0	0	0	0	0
	Config. 3	0	1	1	0	2

Table 6. Defense performance for different configurations

Dataset	**Metric (%)**	**Config. 1**	**Config. 2**	**Config. 3**	**Avg.**
MNIST	Accuracy	90.00	91.67	88.33	90.00
	Precision	83.33	85.71	81.08	83.37
	Recall	100.00	100.00	100.00	100.00
	F1-score	90.91	92.31	89.55	90.92
CelebA	Accuracy	95.83	100.00	91.67	95.83
	Precision	92.31	100.00	85.71	92.67
	Recall	100.00	100.00	100.00	100.00
	F1-score	96.00	100.00	92.31	96.10

5.5 Overall Defense

After evaluating against the defense configurations, the obtained confusion matrices are shown in Fig. 5. The final performance scores are illustrated in Table 6. Because of the flexible values of T_{ssim} and T_{hash}, configuration 2 has the highest detection accuracy, precision, and F1 scores for both datasets, while all configurations have given 100.00% recall scores. An increased recall value ensures that the system has restricted 100% attacks after several iterations. However, false positive is slightly higher in configuration 3 than the others, which indicates more normal users will face restrictions as a result of false detection by our proposed system.

5.6 Results Summary and Discussion

An important measure in this evaluation is the maximum number of iterations before detection by our defense mechanism. A high count of iterations lets the attacker fine-tune the attack model or infrastructure. Fig. 6 and 7 compare the maximum iterations in specific configurations for the two datasets, respectively. It is evident in both Fig. 6 and 7 that configuration 1 and configuration 2 have higher performance scores, despite most labels having a huge maximum iteration count. However, configuration 3 strikes a balance between defense performance and false detection, making it superior in terms of defense. Hence, it is necessary to properly tune the parameters T_{ssim} and T_{hash} to make a trade-off between

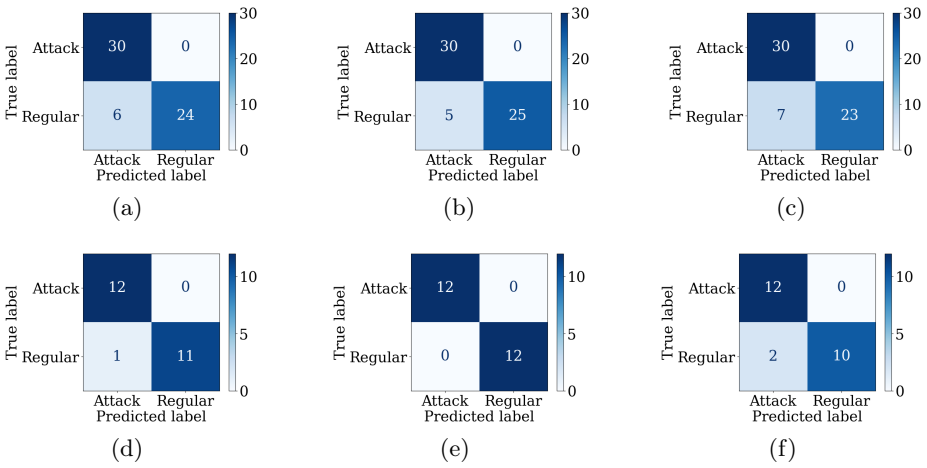

Fig. 5. Confusion matrices for three filter configurations (a)-(c) MNIST and (d)-(f) CelebA

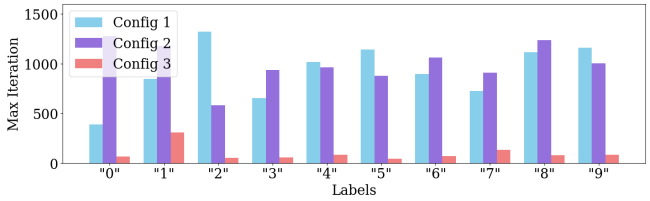

Fig. 6. Comparison of maximum iterations before restriction for MNIST

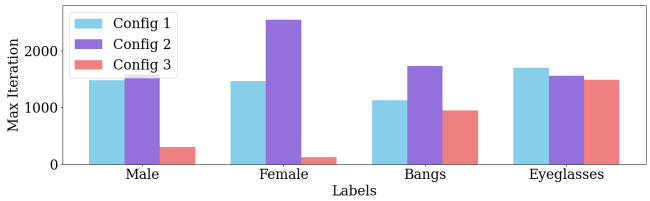

Fig. 7. Comparison of maximum iterations before restriction for CelebA

performance and reliability. For example, setting T_{ssim} too high will make the defense more strict, increasing the number of false positives. Likewise, setting extremely lower values might fail the system in detecting suspicious attacker inputs. These configurations were sufficient for our experiments to exhibit the extent of defense provided by our approach.

6 Conclusion

In this paper, we have proposed a robust filter-based defense strategy that is functional against various MI attacks. Following a series of experiments on two datasets, it is evident that this technique prevents model inversion attacks with remarkable efficacy. It is superior to existing methods in several ways. Firstly, no changes are required in the model architecture to incorporate this method. In addition, it is straightforward to integrate the proposed technique alongside existing trained models located on cloud servers. However, one limitation of this method is tuning the defense configuration properly. Future work includes improving this calibration process and evaluation against multiple attack types on other datasets.

References

1. Carlini, N., Liu, C., Erlingsson, Ú., Kos, J., Song, D.: The secret sharer: Evaluating and testing unintended memorization in neural networks. In: 28th USENIX security symposium. pp. 267–284 (2019)
2. Dibbo, S.V.: Sok: Model inversion attack landscape: Taxonomy, challenges, and future roadmap. In: 2023 IEEE 36th Computer Security Foundations Symposium (CSF). pp. 439–456. IEEE (2023)
3. Fredrikson, M., Jha, S., Ristenpart, T.: Model inversion attacks that exploit confidence information and basic countermeasures. In: 22nd ACM SIGSAC conference on computer and communications security. pp. 1322–1333 (2015)
4. Fredrikson, M., Lantz, E., Jha, S., Lin, S., Page, D., Ristenpart, T.: Privacy in pharmacogenetics: An {End-to-End} case study of personalized warfarin dosing. In: 23rd USENIX security symposium. pp. 17–32 (2014)
5. Gong, X., Wang, Z., Chen, Y., Wang, Q., Wang, C., Shen, C.: Netguard: Protecting commercial web apis from model inversion attacks using gan-generated fake samples. In: Proceedings of the ACM Web Conference 2023. pp. 2045–2053 (2023)
6. Goodfellow, I.: Deep learning. MIT press (2016)
7. Hatamizadeh, A., Yin, H., Molchanov, P., Myronenko, A., Li, W., Dogra, P., Feng, A., Flores, M.G., Kautz, J., Xu, D., et al.: Do gradient inversion attacks make federated learning unsafe? IEEE Transactions on Medical Imaging **42**(7), 2044–2056 (2023)
8. Jegorova, M., Kaul, C., Mayor, C.e.a.: Survey: Leakage and privacy at inference time. IEEE Transactions on Pattern Analysis and Machine Intelligence **45**(7), 9090–9108 (2022)
9. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)
10. Leino, K., Fredrikson, M.: Stolen memories: Leveraging model memorization for calibrated {White-Box} membership inference. In: 29th USENIX security symposium. pp. 1605–1622 (2020)
11. Li, N., Qardaji, W., Su, D., Wu, Y., Yang, W.: Membership privacy: A unifying framework for privacy definitions. In: ACM SIGSAC conference on Computer & communications security. pp. 889–900 (2013)
12. Liu, D., Yang, M., Qu, X., Zhou, P., Cheng, Y., Hu, W.: A survey of attacks on large vision–language models: Resources, advances, and future trends. IEEE Transactions on Neural Networks and Learning Systems (2025)

13. Liu, Y., Wen, R., He, X.e.a.: ML-Doctor: Holistic risk assessment of inference attacks against machine learning models. In: 31st USENIX Security Symposium. pp. 4525–4542 (2022)
14. Liu, Z., Luo, P., Wang, X., Tang, X.: Deep learning face attributes in the wild. In: Proceedings of International Conference on Computer Vision (ICCV) (December 2015)
15. Montavon, G., Samek, W., Müller, K.R.: Methods for interpreting and understanding deep neural networks. Digital Signal Processing **73**, 1–15 (2018)
16. Nasr, M., Shokri, R., Houmansadr, A.: Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In: IEEE symposium on security and privacy (SP). pp. 739–753. IEEE (2019)
17. Pengcheng, L., Yi, J., Zhang, L.: Query-efficient black-box attack by active learning. In: IEEE International Conference on Data Mining (ICDM). pp. 1200–1205 (2018)
18. Rudin, C.: Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature machine intelligence **1**(5), 206–215 (2019)
19. Tramèr, F., Zhang, F., Juels, A., Reiter, M.K., Ristenpart, T.: Stealing machine learning models via prediction APIs. In: 25th USENIX security symposium. pp. 601–618 (2016)
20. Yang, W., Wang, S., Wu, D., Cai, T., Zhu, Y., Wei, S., Zhang, Y., Yang, X., Tang, Z., Li, Y.: Deep learning model inversion attacks and defenses: a comprehensive survey. Artificial Intelligence Review **58**(8), 242 (2025)
21. Yang, Z., Shao, B., Xuan, B., Chang, E.C., Zhang, F.: Defending model inversion and membership inference attacks via prediction purification. arXiv preprint arXiv:2005.03915 (2020)
22. Yang, Z., Zhang, J., Chang, E.C., Liang, Z.: Neural network inversion in adversarial setting via background knowledge alignment. In: ACM SIGSAC Conference on Computer and Communications Security. pp. 225–240 (2019)
23. Yeom, S., Giacomelli, I., Fredrikson, M., Jha, S.: Privacy risk in machine learning: Analyzing the connection to overfitting. In: 31st computer security foundations symposium (CSF). pp. 268–282. IEEE (2018)
24. Zhang, X., Chen, C., Xie, Y., Chen, X., Zhang, J., Xiang, Y.: A survey on privacy inference attacks and defenses in cloud-based deep neural network. Computer Standards & Interfaces **83**, 103672 (2023)
25. Zhou, S., Ye, D., Zhu, T., Zhou, W.: Defending against neural network model inversion attacks via data poisoning. IEEE Transactions on Neural Networks and Learning Systems (2025)
26. Zhu, T., Ye, D., Zhou, S., Liu, B., Zhou, W.: Label-only model inversion attacks: Attack with the least information. IEEE Transactions on Information Forensics and Security **18**, 991–1005 (2022)