PrivMon: A Stream-Based System for Real-Time Privacy Attack Detection for Machine Learning Models

Myeongseob Ko*
myeongseob@vt.edu
Virginia Tech
Blacksburg, VA, USA

Xinyu Yang*
xinyuyang@vt.edu
Virginia Tech
Blacksburg, VA, USA

Zhengjie Ji
zhengjie@vt.edu
Virginia Tech
Blacksburg, VA, USA

Hoang Anh Just
just@vt.edu
Virginia Tech
Blacksburg, VA, USA

Peng Gao
penggao@vt.edu
Virginia Tech
Blacksburg, VA, USA

Anoop Kumar
anooamzn@amazon.com
Amazon
Seattle, WA, USA

Ruoxi Jia
ruoxijia@vt.edu
Virginia Tech
Blacksburg, VA, USA

ABSTRACT

Machine learning (ML) models can expose the private information of training data when confronted with privacy attacks. Specifically, a malicious user with black-box access to a ML-as-a-service platform can reconstruct the training data (i.e., model inversion attacks) or infer the membership information (i.e., membership inference attacks) simply by querying the ML model. Despite the pressing need for effective defenses against privacy attacks with black-box access, existing approaches have mostly focused on enhancing the robustness of the ML model via modifying the model training process or the model prediction process. These defenses can compromise model utility and require the cooperation of the underlying AI platform (i.e., platform-dependent). These constraints largely limit the real-world applicability of existing defenses.

Despite the prevalent focus on improving the model's robustness, none of the existing works have focused on the continuous protection of already deployed ML models from privacy attacks by detecting privacy leakage in real-time. This defensive task becomes increasingly important given the vast deployment of ML-as-a-service platforms these days. To bridge the gap, we propose PrivMon, a new stream-based system for real-time privacy attack detection for ML models. To facilitate wide applicability and practicality, PrivMon defends black-box ML models against a wide range of privacy attacks in a *platform-agnostic* fashion: PrivMon only passively monitors model queries without requiring the cooperation of the model owner or the AI platform. Specifically, PrivMon takes as input a stream of ML model queries and provides an efficient attack detection engine that continuously monitors the stream to detect the privacy attack in *real-time*, by identifying self-similar malicious queries. We show empirically and theoretically that PrivMon can detect a wide range of realistic privacy attacks within a practical time frame and successfully mitigate the attack success rate. Code is available at https://github.com/ruoxi-jia-group/privmon.

CCS CONCEPTS

• **Security and Privacy → Intrusion/anomaly detection and malware mitigation**; • **Computing Methodologies →** *Artificial Intelligence, Machine Learning*.

KEYWORDS

machine learning, privacy, monitoring, detection, label-only attack

ACM Reference Format:
Myeongseob Ko, Xinyu Yang, Zhengjie Ji, Hoang Anh Just, Peng Gao, Anoop Kumar, and Ruoxi Jia. 2023. PrivMon: A Stream-Based System for Real-Time Privacy Attack Detection for Machine Learning Models. In *The 26th International Symposium on Research in Attacks, Intrusions and Defenses (RAID '23), October 16–18, 2023, Hong Kong, China.* ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3607199.3607232

1 INTRODUCTION

Machine learning (ML) has seen unprecedented advancement in a wide range of tasks. However, ML models trained on privacy-sensitive datasets (e.g., medical records [22], face images [48]) can divulge private information through their predictions. Specifically, by simply querying the model through a ML-as-a-service (MLaaS) API [14, 26], an adversary can re-generate a victim's appearance or deduce a victim's medical condition, which leads to privacy leakage.

Membership inference attacks (MFAs) and model inversion attacks (MIAs) are standard *privacy attacks* in the machine learning privacy literature. Both types of attacks exploit the access to a target ML model to infer sensitive aspects about its training data. In particular, MFAs aim to infer whether a victim's data is used for training the target model, whereas MIAs attempt to reconstruct

*Both authors contributed equally to this research.

the training data. Early techniques for both attacks require comprehensive knowledge about the target model (e.g., a neural network's architecture and parameters) or assume that the attacker can query the model to receive the predicted confidence (e.g., the input face image corresponds to Alice with 90% probability). These two threat models, known as *white-box* and *black-box confidence-based*, are often not feasible in real-world settings. The reason is that many real-world MLaaS platforms may neither reveal the inner workings of their models for the sake of intellectual property protection nor provide confidence information for input queries. Instead, these MLaaS platforms only provide predicted labels for input queries, which allows them to maintain their functionality while also preventing the aforementioned attacks from taking effect.

Unfortunately, recent studies have shown that MFAs and MIAs are also carried out in the *label-only black-box setting*, where the attacker can successfully infer private training data by *only receiving predicted labels from the input queries*. Remarkably, the performance of these label-only attacks is on par with and sometimes even better than white-box and confidence-based ones [9, 19]. For example, a label-only MFA can identify the membership with 89.2% accuracy on the CIFAR100 dataset whereas confidence-based MFAs achieve 83%−88% accuracy [9]; label-only MIAs can reconstruct face images of private identities with a success rate of 75.7% [19] on the CelebA dataset, only 7% behind the state-of-the-art white-box attack. Given the effectiveness of the label-only attacks and the practical significance of the underlying threat model, there is an urgent need to mitigate such attacks.

Despite the threat of privacy attacks under the label-only setting, no approach has been specially designed to defend against such attacks in a non-intrusive way. Existing privacy attack defense mechanisms mostly require the cooperation of the underlying AI platform to modify the training procedure to increase the model's robustness, or the prediction procedure to perturb the predictions. Example defenses include implementing differentially private training algorithms [1, 2, 4, 21, 40, 43], adding various regularizers to the training objective [34, 38, 44], and injecting noise into predicted confidence scores during the prediction process [17]. However, these approaches incur significant model accuracy loss and many of them are not designed for label-only settings. Furthermore, these defenses all incur intrusive platform modifications which may be hard to achieve in practice. These limitations greatly undermine the practicality of current defenses.

Contributions. To bridge the gap, in this work, we propose PrivMon, a stream-based system for real-time, platform-agnostic privacy attack detection for ML models. PrivMon is designed to detect different types of label-only privacy attacks in a real-time online fashion, in comparison to most existing works that focus on increasing the model robustness offline. Furthermore, PrivMon operates in the most realistic setting: it is designed to protect black-box ML models in a platform-agnostic fashion, without requiring any modifications to the underlying AI platform and without undermining model accuracies and utilities. PrivMon continuously monitors a stream of queries to the ML model under protection and detects malicious queries that are part of a privacy attack campaign. Next, we describe our key insights in detecting malicious model queries.

PrivMon is inspired by our ***key observation*** of label-only privacy attacks: despite the possible variations in attack goals (MFAs or MIAs) and techniques, existing label-only privacy attacks all share the commonality that they ultimately generate ***a sequence of similar queries***. In particular, label-only MFAs are based on the insight that a model's prediction on a training point is more robust to input perturbations than at a non-training point. Hence, these attacks repeatedly query the target model with small perturbations or transformations of an input query (e.g., adversarial examples, random noise, translation, and rotation) to evaluate the robustness. Additionally, label-only MIAs iteratively form an image reconstruction, which results in a sequence of queries that share similar semantics. On the other hand, benign queries often exhibit natural variations of the real world and are much less similar than malicious ones. Hence, a simple yet effective idea to detect malicious queries is to define the *similarity score* of a query as the average distance between the query and its K most similar past queries, and then mark the query as malicious if the score is high.

However, the task of constructing such a detection system in the real world is non-trivial. There are three main challenges. (1) *Which similarity metric is most suitable for privacy attack detection?* The level of distinguishability between malicious and benign queries highly depends on the choice of similarity metric. In particular, directly using the distance in the pixel space (like the recent work [27]) as a similarity metric would inevitably result in a failure in detecting privacy attacks. Particularly, the transformations (e.g., rotation) used to generate MFA queries can cause a large change in the pixel space, making them indistinguishable from benign queries. In addition, MIAs iteratively optimize a reconstructed image by making a sequence of small modifications in the latent space of a pre-trained neural network generator but there is no guarantee that the small changes in the latent space would result in a small change in the pixel space due to the large Lipschitz constant of neural-network-based generators. Hence, MIAs would also be difficult to detect with the pixel-level distance. (2) *Given a sequence of queries including benign and malicious queries, how to compute the similarity score of each query?* An intuitive approach is to find the K nearest neighbors (KNN) of each query. However, KNN is only acceptable for a small number of queries due to its complexity of $\mathcal{O}(dN + N\log(N))$, where N is the total number of queries and d is the dimension of the input. Given the high number of queries that need to be processed by an MLasS platform in real-time, the low efficiency of such an approach makes it difficult to meet the real-time processing requirement. (3) *How to handle the infinite query stream?* To effectively identify the queries that are most similar to a given query sample, the system should record all prior queries and find the most similar ones among them. Nevertheless, since the queries received by an MLasS platform can be considered an infinite query stream, storing all of them is infeasible.

PrivMon proposes three key designs to deal with the aforementioned challenges: (1) PrivMon leverages neural features extracted from a deep neural network instead of pixel-level information. This design choice is inspired by a study in the computer vision community [47] that the distance between neural features extracted from two images is well correlated with the perceptual similarity between two images. Neural features of an image contain information relevant to differentiating between classes, and therefore, unlike pixel values, neural features tend to be stable to the transformations in MFAs and small latent space modifications in MIAs (Section 4.2). (2) Instead of relying on the naive KNN algorithm, which has a high computational complexity to calculate the distance between neural features, PrivMon innovatively adopts principles from locality-sensitive hashing (LSH) to create a unique encoding scheme for neural features and search the K similar neural features by computing the approximate similarity score. This significantly reduces the computational complexity, making it feasible to process a large number of queries in real-time (Section 4.3). (3) To address the challenge of handling the infinite query stream, PrivMon leverages a sliding window mechanism to record a limited number of historical queries for calculating the score for the current query (Section 4.4). We further provide a formal theoretical analysis of the optimal window size and the attack infeasibility (Section 4.5).

We evaluate the efficacy of PrivMon against four state-of-the-art label-only black-box privacy attacks, including both MFAs and MIAs. We experiment on models trained in a range of datasets that cover standard object detection, street sign recognition, and face recognition tasks. While MFAs are based on adversarial examples and MIAs typically take 4K-5K and 55K queries, respectively, to reach the optimal attack performance, PrivMon detects these attacks after the first 2 queries. Moreover, PrivMon demonstrates remarkable performance in identifying queries associated with an attack (e.g., the Area Under Curve (AUC) Score of $87-95\%$ for transformation-based MFAs and $69-92\%$ for MIAs, and near 100% for the boundary-based MFAs). By rejecting these detected attack queries, PrivMon effectively mitigates privacy attacks (e.g., resulting in an attack success rate close to that of using random queries.) We also investigate possible ways to adaptively attack PrivMon and show that there are simple extensions to PrivMon that make adaptive attacks either suffer from limited attack performance or result in an overly long period to complete an attack.

In summary, this paper makes the following contributions:

- We proposed the design of PrivMon, the first stream-based system for real-time platform-agnostic privacy attack detection for black-box ML models.
- We implemented a functioning prototype of PrivMon and evaluated it against different privacy attacks with various configurations. The results show that PrivMon can effectively detect all these attacks.
- We designed three advanced adaptive attack strategies in which the attacker knows our system design to further evaluate PrivMon's robustness. We find that even faced with advanced techniques, PrivMon can still mitigate the attack success rates.

2 BACKGROUND AND RELATED WORK

At a high level, the goal of privacy attacks is to expose information about private training data through access to a target model. The access could be white-box or black-box. In the *white-box* setting, the adversary has complete knowledge of the target model, whereas, in the *black-box* setting, the adversary is only allowed to make prediction queries against the model. The black-box attacks can be further categorized into *confidence-based* and *label-only* access. In confidence-based attacks, the adversary receives a confidence vector corresponding to the probabilities of the queried input being classified into each possible label class; by contrast, in label-only attacks, the adversary only receives the most likely label for the input query. We will discuss the existing model inversion and membership inference attack techniques for all these different access settings. In addition, we will also review existing defenses.

2.1 Model Inversion Attacks

Model inversion attacks (MIAs) aim at recovering representative training data corresponding to a given output of the target ML model. For instance, an MIA against a face recognition model attempts to recover a representative face image for a given target identity. The key idea behind existing attacks is to solve an optimization problem that seeks an input maximizing the likelihood of producing the given model output.

The first MIA technique was proposed in the context of genetic privacy [13], where the goal was to reconstruct individual genetic markers from access to a personalized medicine model. The proposed attack works for white-box and black-box access but is limited to linear regression and low-dimensional discrete input space. Fredrickson et al. [12] made the first attempt to recover high-dimensional continuous-valued input (e.g., face images), whose idea is to replace exhaustive search with gradient-based search. In particular, a private input is reconstructed iteratively; at each iteration, one calculates the gradient of the likelihood of producing a certain output with respect to the input and then uses the gradient to update the input. This technique demonstrates promising results on simple networks and gray-scale images but becomes completely ineffective for deep networks and RGB images. To address this limitation, most recent works on MIAs [7, 48] leverage generative adversarial networks (GANs) and public data to learn a latent space that produces meaningful images and then solve the optimization problem over the latent space of the GAN instead of the original input space. The proposed techniques in these works need white-box access to the target model. Recently, Kahla et al. [19] introduced a label-only MIA that can reconstruct an input image only based on the knowledge of the corresponding label prediction. To do so, they randomly generate samples on a sphere around the current input and then leverage the label predictions for these samples to estimate the gradient to further solve the optimization problem.

2.2 Membership Inference Attacks

The goal of membership inference attacks (MFAs) is to infer whether or not a given sample is used for training a target model. While only recovering the information about membership in a training set, such attacks still trigger privacy concerns. For example, the knowledge of whether a person's data is used for training a model predicting the dosage for a certain disease can help infer whether the person has that disease. The key idea to enable MFAs is to leverage the overfitting property of ML models—they tend to exhibit different prediction behaviors for training data and those unseen during training.

The first MFA was proposed in [38] under a black-box setting, where authors used a machine learning model (referred to as an attack model) to infer membership of a target sample based on the target model's prediction confidence vector on that sample. To train the attack model, the authors assumed the knowledge of a target model architecture and training data distribution. Follow-up works [16, 29, 33, 37, 39] introduced various techniques to improve the attack performance and further relaxed assumptions made in [38], such as having knowledge of the target model structure or getting a dataset from the same distribution as the target model's training data, to make the attack setting more realistic. Recent studies [9, 30] started to investigate MFAs with label-only access to the target model, which can be categorized into two types: the boundary-based attack and the data-augmentation-based attack. The general idea is based on the observation that if an input query is a member of the training set, it is further away from the decision boundary than a non-member query. Therefore, both studies [9, 30] first estimate the distance from the decision boundary with the adversarial example generation techniques such as "HopSkipJump" [5] and "QEBA" [28], then decide the membership based on the distance to the decision boundary. Note that the process of finding the decision boundary is iterative. Specifically, starting from the target point, one estimates the gradient, uses it to update the point, and repeats these two steps until the point crosses the decision boundary. For the data-augmentation-based attack, the intuition is that the model prediction for a member query is often more stable to transformations such as rotation and translation if the model is trained on those augmentations, and thus one can infer the membership of a query based on the variations of model outputs under input transformations [9]. Specifically, the adversary queries a shadow model with different transformations for a given query to obtain the corresponding labels. The adversary then trains an attack model to learn the association between the labels pertaining to different transformations of a query and the membership of the query. Then, for a given target input, the attacker queries the target model with the transformations and then feeds the received labels to the attack model to determine the membership of the target input.

2.3 Existing Defenses and Limitations

Defenses against privacy attacks. Since successful MFAs rely on overfitting—the model exhibits disparate behaviors on member and non-member inputs—empirical defenses against MFAs are based on reducing overfitting by means of adding $L2$ regularization [38] or adversarial regularization [34]. Differential private (DP) training [1, 2, 4, 21, 40, 43] is another common defense strategy that provides provable privacy guarantees. A DP training algorithm, by definition, produces similar outputs whether or not a training point is included in the training, which precisely aligns with the goal of preventing membership inference. DP training clips the gradient at each iteration and adds noise proportional to the clipping threshold. Despite the formal privacy guarantees, it leads to a low-accuracy model due to the noise injection. In addition, DP-trained models suffer even more severe privacy-utility trade-off when confronted with MIAs, as observed in past works [13, 43, 48]. This is because there might be many inputs (e.g., face images) corresponding to the same output (e.g., identity) in the training set; while DP training hides the presence of a single training point, MIAs can still utilize the association between the input and output learned from the rest of points to reconstruct the input. To effectively defend against MIAs, Wang et al. [44] proposed to employ information bottleneck training to limit the private information memorized in the learned feature representation. The techniques above aim to address the root cause of vulnerability to privacy attacks and thus can potentially lead to better privacy regardless of the attacker's access to the target model (white-box or black-box confidence-based or black-box label-only).

On the other hand, if it is desired to specifically protect against confidence-based attacks, then one can also add noise to the model's prediction scores [17, 36] or reduce their dispersion [46]. But overall, *while there is fruitful research on privacy defenses, the common limitation is that they all require modifying the training algorithm or the model output. Hence, initiating these defenses crucially relies on the cooperation of the ML system operators. At the same time, they suffer from severe utility degradation, limiting their applicability.*

Comparison with similarity-based evasion attack detection. Evasion attacks are a well-studied type of security threat for machine learning models. In these attacks, an adversary adds well-crafted perturbation to a test input so that a trained model misclassifies the test point. Unlike privacy attacks, the goal of evasion attacks is to compromise the functionality of a machine learning model, instead of divulging private training data. Evasion attacks have been developed for both white-box and black-box settings. Recent works Blacklight [27] and Stateful Detection [6] were proposed to defend against black-box evasion attacks. Despite the difference in defense objectives, their core idea is to detect evasion attacks via similarity among queries. However, their techniques will face significant limitations if applied to detecting privacy attacks. In particular, Blacklight [27] calculates the similarity scores based on pixel-space information; therefore, it fails to accurately detect MFAs that rely on transformations (e.g., translation and rotation) and MIAs as they produce queries that have large pixel-space distance to each other.

Stateful Detection [6] trains an encoder with the training dataset of the target model to extract features and calculate the feature distance to all previous queries to find K nearest neighbors. However, two significant drawbacks restrict its practicality. First, it assumes the knowledge of target training data distribution to train the encoder. However, we are considering the practical setting where the model owner does not share sensitive training data distribution even with the defender. Moreover, it performs K nearest neighbor search on entire prior samples, which incurs large runtime and memory. Therefore, it is not applicable to real-time MLaaS operations at scale, as also noted in [27]. Figure 5 and Table 6 in Appendix B clearly demonstrate the limitation of the KNN approach. Therefore, *it is imperative to develop a well-curated real-time system to defend against label-only privacy attacks.* We provide a further comparison with Blacklight in Section 5, and a method that exploits features from a neural network with KNN search (Feature KNN) in Appendix B.

3 THREAT MODEL

We consider privacy attacks against *black-box* ML models, where an adversary can query the target model and access its output. The vast majority of existing attacks utilize confidence scores returned by the target model. However, if the model only displays the predicted label, not the confidence scores, these score-based attacks can be easily countered. Moreover, ML models deployed in user-facing products do not need to reveal the confidence score associated with each class. Hence, in this paper, we focus on *label-only* threat model in which the adversary only receives hard labels when querying ML models. Next, we describe our threat model in detail.

Attacker. An attacker can perform both the MIAs and MFAs against the target model. We denote a target model by $F : \mathbb{R}^{d_{\text{in}}} \rightarrow \mathbb{R}^{|\mathcal{Y}|}$, where $|\mathcal{Y}|$ is the cardinality of the label space $\mathcal{Y}$, and d_{in} is the dimension of the model input. In MIAs, the attacker aims to reconstruct a representative image $x \in \mathbb{R}^d$ from a given target label $y \in \mathcal{Y}$. An MIA is successful if the recovered image x can be recognized as the target label y by a human expert. To scale up evaluation, a classifier is often used in place of a human expert to judge whether the recovered image contains correct semantics. In MFAs, the attacker aims to infer whether a given target input x is in the dataset used for training the target model F. An MFA is successful if it can distinguish members from non-members well. In both label-only MIAs and MFAs, the attacker makes a sequence of queries to launch the attack [9, 19, 29], which are referred to as *malicious queries* hereinafter.

Defender. We assume that the defender can access all queries before they receive responses from the target model. The defender aims at detecting whether a given query is malicious or benign. If the query is deemed benign, the defender returns the corresponding output from the target model. Otherwise, the defender rejects the malicious query or returns random output to mitigate the attack. Moreover, we assume that the defender has limited resources and thus prefers a defense strategy that is memory and computation efficient.

4 THE PRIVMON SYSTEM

4.1 System Overview

To overcome the limitations in existing platform-dependent defenses and bridge the gap, we propose PrivMon, a stream-based system for real-time platform-agnostic privacy attack detection for ML models. Leveraging a stream-based architecture, PrivMon continuously monitors the sequence of queries made to the deployed ML model and detects both label-only MIAs and MFAs in real-time by identifying malicious queries. To facilitate wide applicability and practicality, PrivMon operates in a black-box setting and does not require prior knowledge of the ML model being protected, and is agnostic to the underlying AI platform.

Figure 1 shows the architecture of PrivMon, which consists of two main components: the buffer component and the detector component. PrivMon takes as input a real-time stream of queries made to the ML model being protected and buffers the queries sequentially in a queue (i.e., the buffer component). PrivMon then provides an efficient attack detection engine (i.e., the detection component) that leverages the buffered queries to detect malicious queries. The detection leverages the key observation that malicious queries from a label-only privacy attack tend to be semantically similar, while benign queries are usually dissimilar.

Specifically, for an input query, PrivMon calculates the similarity between the query and the past queries, which is defined as the average distance between the current query and its top-K nearest

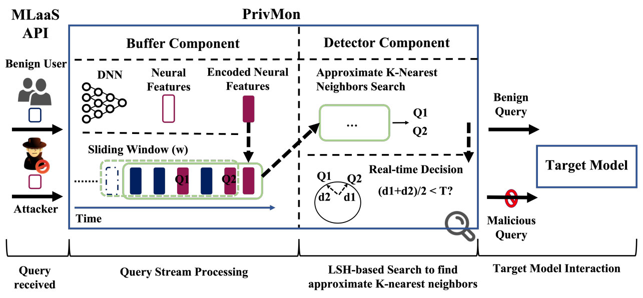


Figure 1: The architecture of PrivMon.

neighbors. If the average distance is smaller than a threshold T, PrivMon considers the current query as malicious and rejects the query before it is sent to the ML model. Depending on the configuration, PrivMon can either return a fake or random prediction to the user who made the query or warns the user with an alert.

Key innovations. To accurately identify self-similar malicious queries and efficiently detect privacy leakage, PrivMon has three key innovations. First, to better capture the semantic meanings of a query, PrivMon maps the query into a neural feature in the hidden space using a deep neural network, such that queries that are semantically similar will have neural features that are close (Section 4.2). Second, to overcome the computation challenge of efficiently searching for K-nearest neighbors in past queries, PrivMon utilizes locality-sensitive hashing (LSH) to encode the neural features. It then selectively sorts past queries, specifically targeting those with hash codes that collide with the current query (Section 4.3). Third, to further reduce the space and time complexity of processing a large number of queries, PrivMon leverages a sliding window to only maintain the most recent queries for efficient LSH search (Section 4.4).

Detection algorithm. Algorithm 1 shows the overall privacy attack detection algorithm employed by PrivMon. For a new input query, PrivMon extracts a neural feature vector (Line 2). Then, PrivMon calculates the hash code for the neural feature vector (Line 3). Next, PrivMon retrieves the K-nearest neighbors *approximately* using LSH (Line 4). Finally, PrivMon compares the average distance between the current query and the K-nearest neighbors with a given threshold (Line 8). If the distance is below the threshold, PrivMon will mark the query as malicious (Lines 9-12). To reduce space and time complexity, PrivMon maintains a sliding window and updates the queries in the window by inserting new queries and evicting old queries (Lines 5-7).

4.2 Neural Query Similarity

PrivMon relies on the key insight that label-only privacy attacks heavily rely on an iterative self-similar query generating process to make a successful attack. However, as some malicious queries can be transformed (e.g., via rotation, or translation) by an adversary, a detection system lacking a robust similarity metric cannot accurately identify the malicious query. To achieve reliable detection performance, a robust similarity metric is necessary.

We found that the perceptual distance metric originally proposed by Zhang et al. [47] is worth considering since it can accurately identify perturbed queries through the lens of neural features extracted

Algorithm 1: Privacy Attack Detection Algorithm

```
In  : current query $x$;
      past queries buffered in the sliding window
      $X = \{x_i : i = 1, \ldots, n\}$;
      hashes of past queries
      $\mathcal{H}(X) = \{\mathcal{H}(x_1), \mathcal{H}(x_2), \ldots, \mathcal{H}(x_n)\}$;
      a pre-trained deep neural network (p-DNN) $\phi$;
      the number of hash functions $R$;
      a family of hash functions $\mathcal{H} = \{h_i : i = 1, \ldots, R\}$;
      sliding window size $w$;
      system threshold $T$;
      the number of nearest neighbors $K$.
Out : Decision $\rightarrow \{0, 1\}$, 0 if $x$ is benign, 1 if $x$ is malicious.
while True do
    Extract neural feature vectors $\phi(x)$ for a given $x$ from a p-DNN $\phi$
    Compute $\mathcal{H}(\phi(x)) = \{h_1(\phi(x)), \cdots h_R(\phi(x)\}$ // compute the hash value with a set of hash functions
    Insert $\mathcal{H}(\phi(x))$ into a set $\mathcal{H}(X)$
    if $card(X) > w$ then
        Delete $\mathcal{H}(x_1)$ from $\mathcal{H}(X)$ where 1 denotes the oldest query
    Find nearest $(d'_1, \ldots, d'_K)$ from $\mathcal{H}(\phi(x))$ to $\mathcal{H}(X)$ via LHS // search $K$ nearest indices with the corresponding distances $d'$ w.r.t $x$
    Calculate $D_{avg} = \frac{1}{K}\sum_{i=1}^{K}(d'_i)$ // calculate the mean distance
    if $D_{avg} > T$ then
        Decision = 0
    else
        Decision = 1
```

from a deep neural network. The perceptual distance measures how similar two images are. The study proposed that the l_2 distance between neural feature vectors extracted from a well-trained neural network highly correlates to human perception. This distance first maps two input queries to corresponding *neural feature vectors*, and then calculates the Euclidean distance between the features. Formally, the perceptual distance between two images x_1 and x_2 can be expressed as

$$d(x_1, x_2) \triangleq \|\phi(x_1) - \phi(x_2)\|_2, \tag{1}$$

where $\phi : \mathcal{X} \rightarrow \Phi$ maps an input $x \in \mathcal{X}$ to the normalized, flattened internal activations of some trained neural network, d represents the dimension of the internal activations and $\phi(x) \in \Phi, \Phi \subseteq \mathbb{R}^d$.

While the perceptual distance metric has been widely used in the computer vision community, our work is the *first* to study the feasibility of this metric in the context of privacy attack detection. The empirical studies in [47] show that the l_2-norm defined on the activations from AlexNet [24] pre-trained with the ImageNet dataset correlates most strongly with human's perception of similarity. In addition, it is preferable to use a model with a minimum number of parameters to extract neural feature vectors due to efficiency concerns. Hence, PrivMon currently leverages AlexNet to extract the neural feature for each query.

In Figure 7, we show that the l_2 distance computed over the input pixel-level information (i.e., without being mapped through a deep neural network) is insufficient to detect malicious queries. On the other hand, neural distances can clearly identify the malicious query if we set the threshold to an appropriate value. This result demonstrates the robustness of the neural distance metric.

4.3 Approximate Nearest Neighbor Search

To determine whether an input query is malicious or not, PrivMon searches through the buffered past queries to check if there exist other perceptually similar malicious queries. This can be achieved with K-nearest neighbors (KNN) search.

KNN search. The KNN search looks for the top-K closest neighbors to the input query. Formally, given a current query point x and the set of past queries D, the set of K-nearest neighbors of x can be defined as $S_x \subseteq D$ such that $|S_x| = K$ and for all $x' \in D \setminus S_x$, $d(x, x') \geq \max_{x'' \in S_x} d(x, x'')$, where d defines a distance metric.

To perform the KNN search for any new query, one needs to sort the *entire* set of past queries, which is expensive when the set of past queries is large and the dimension of x is high. Indeed, KNN search reaches computation complexity $\mathcal{O}(Nd + N\log(N))$, where N is the number of past queries and d is the dimension of the internal activations. In our case, the underlying ML model processes millions of queries per day. This scale makes the exact KNN search method impractical for a real-time detection system.

LSH search. To overcome the computation challenge and improve efficiency, our idea is to avoid sorting the entire history for each new query by leveraging locality-sensitive hashing (LSH) as an approximate KNN search method. LSH is a class of hash functions that generate similar hash codes for similar queries with high probability. We give the formal definition as follows.

Definition 4.1. A family of hash functions $\mathbb{H} = \{h : \mathbb{R}^d \rightarrow \{0, 1\}\}$ is (r_1, r_2, p_1, p_2)-sensitive for $(\mathbb{R}^d, \mathrm{d})$ if for any $x_1, x_2 \in \mathbb{R}^d$ we have

- if $d(x_1, x_2) \leq r_1$ then $Pr_{\mathbb{H}}[h(x_1) = h(x_2)] \geq p_1$,
- if $d(x_1, x_2) > r_2$ then $Pr_{\mathbb{H}}[h(x_1) = h(x_2)] \leq p_2$.

In particular, if the distance between x_1 and x_2 is smaller than r_1, a hash function h will map x_1 and x_2 to the same hash with the probability $\geq p_1$. This demonstrates that LSH can hash two similar neural features into the same hash code with high probability. A locality-sensitive family will be useful if the inequalities $p_1 > p_2$ and $r_1 < r_2$ hold. Here, we use the *random projection* technique to generate hash code [18]. The intuition behind this technique is that if we have one random hyperplane defined by a normal unit vector r, then we can generate the corresponding hash code based on the position of the query relative to the hyperplane. For example, if we have one data point $x = (x^{(1)}, x^{(2)})$ in 2D space, we can draw a random line defined by a norm unit vector $r = (r^{(1)}, r^{(2)})$. Then, we can calculate the dot product between x and r. Taking the sign of the result, we can define a binary hash code for the given data point x. Formally, given an input x and the hyperplane associated with a normal unit vector r, we let $h(x) = \text{sign}(x \cdot r)$, where sign$(\cdot)$ function returns 1 if the argument is positive and 0 otherwise. Geometrically, we can think of whether $h(x)$ is 1 or 0 depending on the angle between the input x and the normal vector r. Using more hash functions (i.e., hyperplanes) can generate more hash codes for a given data point, which enlarges the gap between p_1 and p_2.

We propose to perform LSH on extracted neural features. Specifically, we calculate the hash code for the neural features of an incoming query and only sort the past queries whose neural features correspond to the same hash codes as the incoming query. With this approach, we can circumvent the need of sorting the entire past queries, thereby significantly improving computational efficiency over KNN search by reducing to sublinear complexity $\mathcal{O}(dN^{\rho}\log_{1/p_2}(N))$, where $\rho = \frac{1/p_1}{1/p_2}$ [10].

4.4 Stream-Based System Architecture

Stream processing is a data management technique for rapid real-time processing of *continuous* data streams. Stream-based architectures obtain data from a publish-subscribe service, process the data, and return the result. To enable real-time privacy attack detection, PrivMon employs a stream-based system architecture to manage input queries, extract neural features, and perform LSH search to identify malicious queries in real-time.

Sliding window for query buffering. Since the input queries can be viewed as an infinite stream of data, it is impractical to store all past queries. To further reduce the space and time complexity, PrivMon leverages a sliding window to buffer the most recent queries and uses them as candidates for LSH search.

Specifically, we first initiate a buffer window of size w (e.g., $5K$ queries). Then we collect queries from the query stream and store these queries in the buffer window. When the buffer window is full (i.e., the number of queries is equal to the window size w), we slide the window forward to free space for buffering future queries. Window size and sliding step (i.e., how far to slide forward) are two deciding factors for the effectiveness of the sliding window.

- **Window size:** Intuitively, the more historical information we have, the more likely we are able to detect malicious queries. At the same time, more historical information means more storage occupation and lower processing time, because the current query will be compared with all preserved past queries. In Section 5, we show we can accurately detect most malicious queries of all sequential attacks with small window sizes (e.g., 100 queries). Moreover, we evaluate the system's performance across varying window sizes, with the summarized results presented in Table 13.
  However, for adaptive attacks like a parallel attack (Section 5.5), in which the attacker launches attacks on multiple targets, a larger window is required to cover as many malicious queries for one target as possible. Based on extensive experiments in Section 5.5, we give a recommended window size, $5K$, which can well balance detection effectiveness and system performance.
- **Sliding step:** The selection of the sliding step depends on different scenarios. If the LSH algorithm supports removing samples (e.g., IndexLSH [18]), when the buffer window is full and we receive a new query, we can just remove the oldest query and add the new one. In this case, the sliding step is 1 (the overlapping size is $w - 1$). This approach will keep the buffer window to a constant size.
  On the other hand, if the LSH algorithm does not support removing samples (e.g., IVFIndex with DirectMap of type Array [18]), we have to create a new window and discard the old one. In this case, choosing a good overlapping size matters. Similar to the window size, the larger the overlapping part (the smaller the sliding step), the more information we can preserve. The larger overlapping size means more frequent buffer window renewal and hence more overhead. PrivMon currently leverages IndexLSH with sliding step 1. We leave the exploration of other LSH algorithms to future work.

Enabling LSH search over the stream. Since we maintain a buffer of window size w, we are able to perform the LSH search within the buffer. As LSH search is an approximate search approach, it would achieve high efficiency by sacrificing search accuracy to some extent. To achieve a higher accuracy without increasing overhead, we propose a *two-level scheduling*. Specifically, for an incoming query, we use LSH to calculate its hash code and assign it to the corresponding bucket; the hash codes of neighboring queries in the buffer should collide with that of the incoming query. Then, we use the exact KNN algorithm to find the K nearest neighbors from the same bucket. The combination of these two steps allows us to achieve high efficiency while preserving stable accuracy for the stream.

4.5 Formal Analysis

We will present a formal analysis showing the strength of protection provided by PrivMon. In particular, we analyze at worst case how many malicious queries would fall into the same window for any given window size, a desired attack completion time, and the total number of queries needed to enable a successful attack. Note that if only a single malicious query fell into the same window, the nearest neighbor search would be ineffective; and the more malicious queries an attack issues, the higher chance that the attack gets detected by the nearest neighbor search. We show that for reasonable choices of attack completion time and window sizes that remain computationally efficient, there must be a large number of malicious queries falling into the same window, thereby enabling effective detection via nearest neighbor search.

Proposition 4.2 (Attack Infeasibility Analysis). *Let S be a detection system with a sliding window of size w that processes Q number of queries per day. Consider an attacker which requires at least $\mathcal{M}$ number of malicious queries for a successful deployment of an attack within T days. Then, there exists a window in which there are at least $\left\lceil \frac{w}{\left\lceil \frac{Q \cdot T}{\mathcal{M}} \right\rceil} \right\rceil$ number of malicious queries.*

We can now apply the result of Proposition 4.2 to the real case scenario. Let S be our detection system with a window size $w = 5K$ that can process $Q = 1M$ queries per day. Consider an attacker performing the MFA [29] attack on our system. For a successful attack, the attacker will need to query the target model with $|\mathcal{M}| = 50K$ malicious points within a reasonable time, say $T = 30$ days. By Proposition 4.2, we calculate that there exists at least one window in which there are at least 9 malicious queries. However, we note that our detection system is highly effective and it can accurately detect malicious queries. Specifically, for any MFA attack, as shown in Table 11, PrivMon is able to detect malicious queries via K-nearest neighbor search even with only 2 initial malicious queries. Therefore, the attacker would need to decrease the number of malicious queries within a single window to at most 1 to avoid detection. This however would increase attack time to at least $T = 250$ days, which deems the attack highly impractical.

5 EVALUATION

5.1 Experimental Setup

In this section, we evaluate PrivMon on a variety of label-only privacy attacks. We expatiate the defense settings, datasets, and system evaluation metrics employed. To demonstrate the performance of our defense algorithm, we incorporate datasets and model architectures from previous literature [9, 19, 29] and further extend with more comprehensive attack scenarios.

Attacks. We consider four state-of-the-art label-only privacy attacks: (1) "HopSkipJump"-based MFA **(HSJA)** [30], which discovers the membership of a target point based on the distance of the point to the decision boundary of the target classifier: non-training points are observed to be closer to the decision boundary than training points. Computing this distance is exactly the problem of finding the smallest adversarial perturbation, which can be done using label-only access to a classifier. HSJA leverages the HopSkipJump algorithm proposed in [5] to find the smallest perturbation. (2) "Query-Efficient Boundary"-based MFA **(QEBA)** [30] shares the same idea as HSJA of using the smallest adversarial perturbation to discern membership information. However, it utilizes the QEBA algorithm proposed in [28] to more efficiently calculate the adversarial perturbation. (3) Data augmentation-based MFA **(DA)** [9] that generates malicious queries by transforming a target point via translation and rotation and evaluates the target model's robustness to transformation. (4) **BREPMI** [19] is the most recent label-only MIA. It requests the model's predicted labels over a sphere and then estimates the direction to reach a target class's centroid.

We consider 200 target points with 50 and 150 iterations for HJSA and QEBA respectively, 100 target points for BREPMI with 1000 iterations, and 1000 target points for DA. In the case of transformations for DA, we employ either translation (denoted as d) or rotation (denoted as r). Specifically, we opt to use a rotation magnitude of $r = 5$ and a translation magnitude of $d = 1$. With these parameters, we generate $N = 2r + 1$ rotated images by a magnitude $\pm r$ and $N = 4d + 1$ translated images with a pixel bound of d (such that $|i| + |j| = d$) where horizontal and vertical shifts of $\pm i$ and $\pm j$, respectively, were applied. The average number of queries for each attack and the number of target samples are summarized in Table 12.

We would like to clarify that we empirically choose a single threshold throughout our experiments. We provide the results of threshold selection in Figure 6 which can be found in Appendix C.

Defense. Our system uses a universal threshold T as 350 over different attacks and for different datasets. Additionally, we utilize the window size w as $5K$ throughout all experiments. The selection of window size is solely dependent on the defender's budget, considering the trade-off between the storage and time complexity. We present an ablation study of the window size selections in Appendix D. Moreover, we choose $R = 2048$ to generate a hash code for each data and set $K = 2$ for searching approximate K-nearest neighbors. Our perceptual model is based on AlexNet [24] which is pre-trained on ImageNet dataset [11].

Baselines. *It is worth noting that there is no direct baseline to compare with since this is the first work aiming at detecting privacy attacks for black-box hard-label-output machine learning models.* However, we consider the following two baselines. The first baseline is the state-of-the-art but was originally designed for detecting evasion attacks but shares a similar idea of identifying similar queries. The second baseline was designed to understand the advantage of neural features over the input pixels for similarity comparison. We further present a comparison with the KNN search in the Appendix.

- **Blacklight:** The first baseline is the state-of-the-art evasion detection system proposed by [27]. We re-purpose it for privacy attack detection.
- **Input LSH:** The second baseline is when a defender does not use deep features and instead relies on input pixels with LSH.

Datasets. We consider the CIFAR10 and CIFAR100 datasets [23] to evaluate our proposed MFAs. Furthermore, we assess the performance of HSJA and QEBA-based MFAs on the GTSRB dataset [42]. It is important to note that we exclude the GTSRB dataset for DA evaluation due to its poor attack performance. For BREPMI, we follow the prior literature and consider the two face datasets, namely, CelebA [31] and FaceScrub [35]. The details about each dataset are described in Tables 7 and 8 in Appendix F.

Evaluation metrics. If PrivMon detects a malicious query, the system will reject the query to prevent the attack completion and the attacker will stop sending the subsequent malicious queries. We use the following metrics to evaluate our system performance.

- Detection Success Rate (**DSR**). The DSR describes the detection performance of an underlying system. In particular, it represents the detection performance on a sequence of malicious queries combined with benign queries. We use the Area Under the ROC Curve (AUC) score to evaluate the performance. As we consider two classes (i.e., benign or malicious), the random guess performance is 0.5 (i.e., randomly selecting between benign and malicious queries).
- Attack Success Rate (**ASR**). The ASR indicates the attack performance. Many prior works mentioned in [3] on MFAs usually report average-case success metrics (e.g., accuracy or AUC score) to demonstrate the performance over all target points (i.e., the fraction of the target point whose membership information can be correctly inferred by an attack). Similarly, the prior studies on MIAs [7, 19] also use the accuracy metric (i.e., the fraction of target classes for which an attack can successfully reconstruct the corresponding features) for evaluation. For MFAs, the random guess baseline's performance is 0.5 (i.e., randomly selecting between "member" and "nonmember"). While, the random baseline's ASR for MIAs is 0, since for each target class random pixels are generated as reconstructions. An ideal defense would reduce the ASR of various attacks to the ASR of their corresponding random baselines.

- False Positive Rate (**FPR**). The FPR is the proportion of benign queries incorrectly identified as malicious ones.
- CPU Usage: The percentage of CPU occupied by the system running.
- Memory Usage: The memory occupied by the system running.
- Processing Time: The average processing time of one query.

5.2 System Effectiveness in Attack Detection and Mitigation

5.2.1 Attack Detection. We commence our evaluation by assessing the efficacy of PrivMon in detecting the whole sequence of malicious queries interwoven with benign ones. We assume the attacker is persistent and continues sending subsequent queries to complete an attack even after being detected. Table 1 shows the evaluation results of PrivMon and the two baselines for different attacks on various datasets.

As demonstrated in Table 1, PrivMon exhibits superior performance to all baselines across all attacks and datasets. For malicious queries from HSJA and QEBA, all detection systems achieve high detection performance (e.g., $0.995 \sim 1.000$) on both CIFAR10 and CIFAR100 because of the high self-similarity between queries. The superiority of our system is more pronounced in the case of DA and BREPMI. For example, Blacklight's DSR drops significantly to around $0.501 \sim 0.550$, when faced against different types of DA. The degradation in DSR performance for Input LSH in the face of DA is less severe on both the CIFAR10 (e.g., 0.772 and 0.874) and CIFAR100 (e.g., 0.779 and 0.879) than Blacklight as this method takes advantage of all input information for similarity search without sampling. The query detection performances for BREPMI are generally lower than those for HSJA and QEBA since BREPMI generates images whose semantic distances are getting farther away from each other during the optimization process. Nonetheless, our system can still achieve the best DSR performances on both FaceScrub on CelebA datasets, with $0.029 - 0.173$ higher on FaceScrub and $0.151 - 0.398$ higher on CelebA, compared to other baselines. Our results indicate that Blacklight can solely attain performance levels akin to random guessing for MIA queries. In addition, the substantial decrease in performance from the FaceScrub dataset can be attributed to the low quality of the malicious queries utilized during optimization, compared with the CelebA dataset, consequently leading to a lower attack success rate [19].

Takeaways. Our experimental results suggest that leveraging pixel-level information (e.g., Blacklight, Input LSH) for similarity search is effective for evasion-based MFAs since they are designed to detect evasion attacks. However, these approaches fail to obtain high DSR for other types of privacy attacks, including BREPMI and DA, as their corresponding malicious queries have a large pixel-level distance from one another. *These results justify the use of neural feature-based similarity measure.*

5.2.2 Attack Mitigation. In this part, we scrutinize the real attack and defense cases to evaluate whether PrivMon can effectively reduce the ASR while maintaining a low false positive rate. In this experiment, an adversary cannot proceed to complete the attack and has to stop querying if PrivMon does not return a label to the input query.

Table 1 shows the summarized evaluation results. We can observe that PrivMon effectively mitigates the ASR for all types of privacy attacks on diverse datasets. In particular, for HSJA and QEBA, all methods attain high mitigation performance, thereby allowing the attacker to achieve an ASR of around 0.500 (i.e., random guess). This is because these attacks generate highly self-similar queries at the input level. In the case of QEBA, the attacker has to access multiple queries to the target model to receive a label and estimate the initial gradient. Therefore, if we set $K = 2$, PrivMon blocks the queries from the attacker in an early stage. The number of malicious queries our system has to accept before the initial detection, leveraging our neural query similarity, is shown in Table 11.

The key advantage of PrivMon arises from considering queries that are semantically similar but dissimilar in the input space. As delineated in Table 1, PrivMon significantly diminishes the ASR of DA to nearly 0.535 from 0.793 and to 0.500 from 0.808 on CIFAR10 with different augmentations. Similarly, PrivMon can lower the ASR to 0.739 and 0.573 from 0.916 and 0.938 on CIFAR100, respectively, while sustaining a lower false positive rate. The mitigation performance from Blacklight is quite limited compared to that of our system (0.784, 0.736 on CIFAR10, and 0.917, 0.885 on CIFAR100). These results coincide with the results demonstrated in DSR in the sense that Blacklight is inadequate at detecting malicious queries related to DA. The reason is that malicious queries are semantically similar but not similar in the input domain; therefore, the pixel-level information sampled from an image is not sufficient to correctly mark malicious queries. Similarly, Input LSH is also sensitive to input variations, as suggested by the high FPR.

Regarding BREPMI in Table 1, all methods can achieve high mitigation performance. PrivMon can significantly reduce the attack's performance to nearly 10% from 65% and 89% on FaceScrub and Celeba datasets, respectively, with a negligible false positive rate. The mitigated ASR (10% and 12%) with PrivMon are higher than the mitigated ASR (1% and 4%) on the FaceScrub and CelebA datasets using Blacklight, respectively, with a comparable false positive rate. However, it is worth mentioning that if the attacker achieves the attack success rate near 10%, the attacker cannot derive meaningful information from reconstructed images as shown in Figure 2. In Figure 2, the fourth column (reconstructed images without PrivMon) to the fifth column (reconstructed images with PrivMon) illustrates that PrivMon successfully precludes the attacker from rebuilding privately meaningful information about the target identity.

These results may not be strongly correlated with the results from DSR. This observation arises from the fact that in the initial stages of the BREPMI, the malicious queries are very similar to one another due to the small radius utilized in the gradient estimation step, which subsequently leads to high mitigation performance for other baselines. However, this advantage can be easily nullified if the attacker simply excludes similar samples to evade protection systems in the early stage. If the attacker can evade the detection system for a few steps, it becomes increasingly challenging to detect malicious queries because of the large distances between samples; as shown in Table 1, Blacklight and Input LSH cannot achieve high

Table 1: Detection and mitigation performance comparison.

	Attack Type	Dataset	Detection				w/o	Mitigation						
			Random	Input LSH	Blacklight	PrivMon	Defense	Random	Input LSH		Blacklight		PrivMon	
			DSR	DSR	DSR	DSR	ASR	ASR	ASR	FPR	ASR	FPR	ASR	FPR
MFAs	HSJA	CIFAR10	0.500	**1.000**	0.985	**1.000**	0.852	0.500	0.519	0.000	**0.500**	0.000	**0.500**	0.000
		CIFAR100	0.500	0.999	0.995	**1.000**	0.955	0.500	0.586	0.006	**0.500**	0.000	**0.500**	0.000
		GTSRB	0.500	**0.999**	0.993	**0.999**	0.924	0.500	0.538	0.007	**0.500**	0.000	**0.500**	0.018
	QEBA	CIFAR10	0.500	0.999	0.995	**1.000**	0.917	0.500	**0.500**	0.000	**0.500**	0.000	**0.500**	0.000
		CIFAR100	0.500	**1.000**	0.995	**1.000**	0.966	0.500	0.610	0.004	**0.500**	0.000	0.509	0.000
		GTSRB	0.500	**0.999**	0.993	**0.999**	0.965	0.500	0.504	0.001	**0.500**	0.000	**0.500**	0.004
	DA ($d1$)	CIFAR10	0.500	0.772	0.502	**0.882**	0.793	0.500	0.582	0.147	0.784	0.019	**0.535**	0.019
		CIFAR100	0.500	0.779	0.501	**0.879**	0.916	0.500	**0.731**	0.165	0.917	0.020	0.739	0.045
	DA ($r5$)	CIFAR10	0.500	0.874	0.548	**0.948**	0.808	0.500	0.518	0.138	0.736	0.020	**0.500**	0.016
		CIFAR100	0.500	0.879	0.558	**0.951**	0.938	0.500	0.759	0.153	0.885	0.021	**0.573**	0.038
MIAs	BREPMI	FaceScrub	0.500	0.668	0.524	**0.697**	0.650	0.000	**0.010**	0.086	0.060	0.000	0.100	0.001
		CelebA	0.500	0.775	0.528	**0.926**	0.890	0.000	**0.040**	0.005	0.120	0.000	0.120	0.000

Table 2: A comparative analysis of system overhead among three systems.

Attack Type Dataset	Metric	Memory Usage (MB)		CPU Usage (%)		Throughput (#/Day)
		10K-mean	10K-std	10K-mean	10K-std	
HSJA CIFAR10	PrivMon	2690	12.6	26.3	1.0	2.74 M
	Input LSH	2558	0.4	28.3	0.6	2.74 M
	Blacklight	2455	3.3e-13	11.4	0.3	2.52 M
HSJA GTSRB	PrivMon	2618	0.9	15.7	0.1	554 K
	Input LSH	2554	1.5	18.2	0.5	908 K
	Blacklight	2306	6.6e-13	11.3	0.2	2376 K
QEBA CIFAR10	PrivMon	2686	0.6	28.5	1.0	2.24 M
	Input LSH	2677	0.3	33.1	1.1	2.74 M
	Blacklight	2455	6.6e-13	11.3	0.2	2.51 M
QEBA GTSRB	PrivMon	2354	3.6	16.0	2.2	542 K
	Input LSH	2575	18.0	18.4	0.5	870 K
	Blacklight	2306	6.6e-13	11.2	0.1	2377 K
MIA FaceScrub	PrivMon	2620	2.1	15.5	0.2	546 K
	Input LSH	2562	0.7e-3	18.5	0.5	883 K
	Blacklight	2363	24.8	11.4	0.2	2436 K
DA ($d1$) CIFAR10	PrivMon	2554	19.0	28.4	0.2	2.39 M
	Input LSH	2540	17.0	31.5	1.1	2.97 M
	Blacklight	2455	4.6	11.2	0.2	2.52 M
DA ($r5$) CIFAR10	PrivMon	2674	19.0	24.0	1.1	2.35 M
	Input LSH	2680	18.9	28.4	0.2	3.12 M
	Blacklight	2455	6.6e-13	11.0	0.2	2.52 M

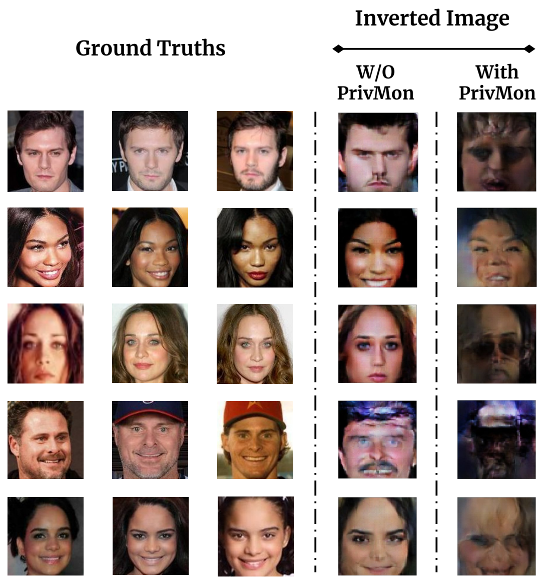


Figure 2: Selected examples of the impact of PrivMon on model inversion attack image generation. As shown from the fourth column, without PrivMon, an adversary can reconstruct meaningful images. On the other hand, as described in the fifth column, the adversary cannot achieve meaningful information from the target model protected by PrivMon.

DSR as ours. In Section 5.5, we consider the possibility of excluding similar samples to evade our system.

Takeaways. In summary, PrivMon can effectively mitigate ASR for all types of attacks on all datasets while preserving a low FPR. The use of neural feature information is crucial in capturing semantic similarities, particularly for queries that are dissimilar in input space. *Therefore, it is imperative to consider the robust metric in privacy attack detection.*

5.3 System Overhead

To extensively evaluate the overhead of PrivMon, we evaluate the memory usage, CPU usage, and throughput of PrivMon with different metrics for detecting different types of attacks with different datasets. We perform all the evaluations on a server with an Intel(R) Xeon(R) Gold 5115 CPU @ 2.40GHz, an NVIDIA Tesla P40 GPU, and 38GB memory. Detecting different datasets with the same image size usually has the same system overhead, so we evaluate datasets with different image sizes for each attack case. In the end,

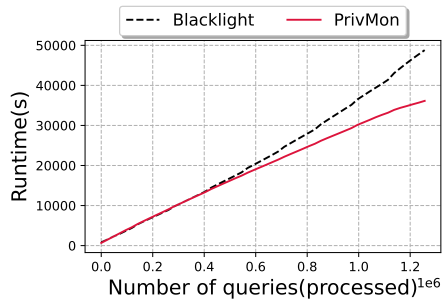


Figure 3: Runtime comparison between Blacklight and PrivMon. Our system outperforms Blacklight in processing speed.

we evaluate four attacks: BREPMI attack for the FaceScrub dataset, HSJA and QEBA attack for CIFAR10 and GTSRB datasets, and DA attack for the CIFAR10 dataset. Similar to Section 5.2, PrivMon and two baselines are evaluated (the window size is 5K). The results are shown in Table 2. In this experiment, we leverage 10K queries to calculate the Memory Usage, CPU Usage, and Throughput.

We have the following observations: 1) PrivMon takes no more than 3GB memory for all attacks and metrics. We believe such memory occupation is negligible for modern servers. 2) PrivMon occupies less than 29% of CPU for all evaluated scenarios, which is only a small part. 3) PrivMon is able to process more than 540K queries for 64x64 images and more than 2.2M queries for 32x32 images per day. 4) The standard deviation of our evaluation is relatively small, which means our evaluation results are representative.

Runtime Comparison between PrivMon and Blacklight. We additionally present runtime complexity comparison between PrivMon and Blacklight when both systems receive around $1.2M$ queries. Since Blacklight does not provide any technical query deletion method, the runtime complexity will increase if the number of queries increases. Considering the result in Figure 3, the number of queries PrivMon can process for one day is much higher than Blacklight as our system processed $1.2M$ queries within 11 hours, but Blacklight took nearly 14 hours.

Takeaways. Thus, as mentioned in Section 2.3 and [27], the Feature KNN approach is not applicable in real-world MLasS, considering the high processing time, memory usage, and CPU usage. The sliding window design and the support of neural LSH search help PrivMon to manage the computation and memory efficiency. *Therefore, all components of PrivMon are closely related to improving the system's search robustness.*

5.4 Video Data

Another limitation of Blacklight, as discussed in [27], is that it considers a sequence of video frames as benign queries. Though it might be a desirable behavior, however, this poses another question in privacy attacks; *what if an attacker can exploit a sequence of video frames (semantically similar) as an alternative to data augmentations to assess the membership information?* As shown in Table 3, we empirically demonstrated that attackers could exploit video frames to achieve a highly successful attack (with accuracy ≈ 0.90). To protect against this attack, a desirable system should mark the semantically

Table 3: Left: The attack success rate (ASR) of data augmentation-based MFA using different video datasets. Random indicates the random guess performance (i.e., member or nonmember), and $\mathcal{M}1$ and $\mathcal{M}2$ represent different model architectures. Right: The detect success rate (DSR) on the part of video frames.

Dataset	Random	$\mathcal{M}1$	$\mathcal{M}2$
	ASR	ASR	ASR
YouTubeFaces [45]	0.500	0.911	0.866
UCF101 [41]	0.500	0.912	0.880

Datasets	Blacklight	PrivMon
	DSR	DSR
YouTubeFaces [45]	0.750	0.830
UCF101 [41]	0.698	0.930

Table 4: PrivMon performance on parallel DA attacks.

Dataset [Parameter]	Original Window Size(5K)			Adaptive Window Size(100K)		
	ASR	FPR	FNR	ASR	FPR	FNR
CIFAR10 ($d1$)	0.778	0.023	0.980	0.521	0.050	0.299
CIFAR10 ($r5$)	0.811	0.0.017	0.927	0.500	0.057	0.088
CIFAR100 ($d1$)	0.907	0.089	0.954	0.650	0.089	0.281
CIFAR100 ($r5$)	0.905	0.040	0.898	0.578	0.098	0.086

similar queries as malicious. As can be observed through DSR in Table 3, PrivMon achieves better malicious query detection performance on a portion of video frames than Blacklight. The detailed setup of the video experiments is demonstrated in Appendix E.

5.5 Adaptive Attacks

In this section, we present an evaluation of PrivMon's effectiveness against three advanced attackers possessing prior knowledge of the target model and all related hyper-parameters.

5.5.1 Adaptive Attack Setup. In the adaptive attack evaluation, due to the high computational complexity involved in generating perceptually dissimilar images, we reduce the number of target points and iterations for HSJA and BREPMI. Specifically, the number of target points is reduced from 200 to 100 and from 100 to 50, respectively. As QEBA is an efficient version of HSJA, we only consider HSJA for MIA. In the case of DA, we increase the number of data points from 1000 to 2500 to account for the parallel attack scenario.

When generating the perceptually adversarial images, the attacker needs a local classifier C trained on the same data distribution. The advanced attacker leverages pre-trained ResNet50 [15] and FaceNet as local classifiers for HSJA and BREPMI to generate perceptually dissimilar queries, respectively, and considers different attack budgets (e.g., $\epsilon \in [0.5, 1.0, 1.5, 2.0]$) to evade our system. Finally, the attacker also uses the same perceptual model as our system.

5.5.2 Parallel Attacks. An advanced attacker of DA knows that PrivMon is using the sliding window technique, the advanced DA attacker sends a single malicious query per each window from multiple target points simultaneously. Assuming that PrivMon's window size is $5K$ and PrivMon maintains $1 : 1$ ratio between

Table 5: PrivMon performance on two types of advanced attackers: 1) The attacker can exclude perceptually similar queries, and 2) the attacker can generate perceptually dissimilar queries.

Attack Type		Dataset	Random	w/o Defense	PrivMon														
					case 1			case 2 (ϵ = 0.5)			case 2 (ϵ = 1.0)			case 2 (ϵ = 1.5)			case 2 (ϵ = 2.0)		
			ASR	ASR	ASR	FPR	FNR	ASR	FPR	FNR	ASR	FPR	FNR	ASR	FPR	FNR	ASR	FPR	FNR
MFAs	HSJA [9, 29]	CIFAR10	0.500	0.860	0.500	0.000	1.000	0.560	0.000	1.000	0.710	0.002	1.000	0.670	0.010	1.000	0.670	0.010	1.000
		CIFAR100	0.500	0.970	0.500	0.000	1.000	0.500	0.000	1.000	0.810	0.020	1.000	0.760	0.020	1.000	0.760	0.020	1.000
MIAs	BREPMI [19]	FaceScrub	0.002	0.700	0.160	0.000	1.000	0.140	0.000	1.000	0.140	0.020	1.000	0.140	0.050	1.000	0.120	0.060	1.000
		CelebA	0.001	0.900	0.010	0.000	1.000	0.120	0.001	1.000	0.100	0.010	1.000	0.120	0.010	1.000	0.120	0.010	1.000

benign and malicious samples if the attacker sends a single malicious query from $2.5K$ data points simultaneously, the next batch of malicious queries for these data points will be considered benign samples. Therefore, as shown in Table 4, PrivMon cannot effectively mitigate the attack performance (e.g., high False Negative Rate (**FNR**)). In this case, PrivMon can still mitigate the attacker's performance while maintaining the low false positive rate (< 0.1) by increasing the window size ($w = 100K$). If the attacker wants to evade PrivMon more than the current window size, the attacker has to spend much more time to complete the attack (e.g., $23 - 24$ queries per each day, $(27500/23) \approx 1196$ days). We also evaluate the detection performance under the parallel attack scenario. In Table 10 in Appendix B.3, we can see that PrivMon still achieves the best performance for parallel adaptive attacks.

Takeaways. *If the attacker wants to launch the parallel scenario, the attacker has to spend a lot of time, even for a small window size ($w = 5K$).* For example, in the case of BREPMI, the attacker has to spend almost 514 days to complete just one attack, given the processing capacity and the required number of queries.

5.5.3 Excluding Perceptually Similar Queries. In the second scenario, an advanced attacker wants to evade PrivMon by considering PrivMon's system design. Specifically, the attacker calculates the neural distance between queries using the same strategy as ours and excludes the perceptually similar samples to evade PrivMon. The evaluation results are presented in case 1 of Table 5. We can observe that the attacker can only achieve an accuracy of 16% and 10% for BREPMI on each dataset and obtain a performance close to random guess (~ 0.500) for HSJA on CIFAR10 and CIFAR100. The reason is that if the attacker excludes perceptually similar samples, the attacker can only rely on limited samples to estimate the gradient direction in BREPMI, which leads to the failure of finding the optimal point. Similarly, in HSJA, the initial queries are already perceptually similar to each other, rendering the attacker unable to proceed to the boundary step if they exclude perceptually similar samples, ultimately leading to poor attack performance.

5.5.4 Generating Perceptually Dissimilar Queries. In the third scenario, the most sophisticated attacker can use the perceptual adversarial training method [25] to generate perceptually dissimilar queries to bypass our system. As outlined in [25], the attacker can generate the perceptual adversarial images as follows:

For a given input x with a true label y, an advanced attacker wants to generate a perceptual adversarial image $\tilde{x}$ with a budget ϵ to make a model $C : \chi \rightarrow Y$ misclassify:

$$C(\tilde{x}) \neq y,\ d(x, \tilde{x}) = \|\phi(x) - \phi(\tilde{x})\| \leq \epsilon \tag{2}$$

$$\max_{\tilde{x}} L(C(\tilde{x}, y)) - \lambda \cdot max(0, \|\phi(\tilde{X}) - \phi x)\| - \epsilon) \tag{3}$$

The perceptual constraint cost is designed to be 0 as long as the generated perceptual adversarial sample is within the perceptual distance ϵ.

In this case, the highly skilled attacker generates perceptually distinct queries whenever querying PrivMon is required. Concurrently, the attacker can rule out the samples that are perceptually similar to the previous query history. In particular, for BREPMI, the attacker generates multiple samples using the trained generator for a given initial point and then transforms those samples into perceptual adversarial queries. Following that, the attacker excludes the perceptually similar samples, considering the query history, and repeats this process to estimate the gradient. In a similar manner, for HSJA, the attacker produces perceptually dissimilar queries in the binary, gradient estimation steps since each step requires multiple self-similar queries. At the same time, the attacker excludes perceptually similar queries, considering the query history.

The results in case 2 from Table 5 show that the attacker generally achieves higher performance than 0.500 for MFA if the attack budget used is greater than 1.0. For BREPMI, the attacker cannot significantly improve the attack performance because the generated samples are farther apart from each other and make it difficult to be in the uniform sphere, resulting in poor gradient estimation. In most of the cases, as we expected, the system has $FNR = 1.000$ since the malicious samples are designed to evade PrivMon.

Takeaways. Skillful attackers who have knowledge of our system design can attempt the parallel attack, exclude the perceptually similar queries or generate perceptually dissimilar queries. *However, in most cases, PrivMon can still effectively mitigate the ASR or requires the attack to spend a lot of time to complete the attack.*

6 FURTHER DISCUSSION AND CONCLUSIONS

In this paper, we propose PrivMon, a platform-agnostic real-time privacy attack detection system. Considering the existing challenges in the literature, we design three technical aspects. First, we leverage neural feature information to accurately identify self-similar malicious queries. Second, to enable the real-time search of nearest neighbors, we leverage LSH to approximate the KNN search. Furthermore, considering the large number of daily queries that our system receives, we discard the old queries by using a sliding window technique for practical storage and search time complexity, as described in Section 4. Compared with current baselines, PrivMon is effective and efficient in terms of performance (e.g., detection and mitigation performances) and efficiency (e.g., CPU usage, memory usage, and search time). Moreover, we consider the three advanced attackers and show that PrivMon can either mitigate the attack performance or require the attacker to spend near infeasible time to complete the attack. Two recent works [6, 27] related to our paper have their limitations to be effective in defending against label-only privacy attacks.

Limitations of our system design. PrivMon currently only focuses on image-related privacy risks and it leverages the neural feature LSH as a search metric. However, both MIAs and MFAs can occur in the natural language domain as well, and we leave this for future research. Another limitation is that when multiple benign users send similar benign samples within the window size, PrivMon indeed rejects the benign queries from benign users since the system marks it as a potential malicious attack (false positive). However, we believe that this seldom happens since it is difficult to have very similar benign queries (or the same benign queries) from many benign users within the limited window size. We discuss the potential solution to addressing this limitation in the following design alternatives.

Design alternatives. Instead of storing all the sequential queries, we can devise a selective-saving approach to preserve the statistically meaningful benign and malicious queries within the window size. In particular, if some potential queries are self-similar (e.g., within a distance boundary) to one of the previous queries, we do not have to save those queries because it is statistically overlapped. Another design alternative would be dynamically changing the sliding window size. If we fix the window size, we always need to search over w queries. However, if we start from a much smaller window, such as 2, and exponentially increase the sliding window size (when there is no malicious query within the window size), we can have a chance to detect malicious queries for a not-so-knowledgeable attacker. The advantage of this design is that it can cover any parallel attack, but the disadvantage is that it incurs higher computational complexity if the attacker sends malicious queries very sparsely.

REFERENCES

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

[2] Raef Bassily, Adam Smith, and Abhradeep Thakurta. 2014. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*. IEEE, IEEE, Washington, DC, 464–473.

[3] Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramèr. 2022. Membership Inference Attacks From First Principles. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, USA, 1897–1914. https://doi.org/10.1109/SP46214.2022.9833649

[4] Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. 2011. Differentially private empirical risk minimization. *Journal of Machine Learning Research* 12, 3 (2011).

[5] Jianbo Chen, Michael I Jordan, and Martin J Wainwright. 2020. Hopskipjumpattack: A query-efficient decision-based attack. In *2020 ieee symposium on security and privacy (sp)*. IEEE, 1277–1294.

[6] Steven Chen, Nicholas Carlini, and David Wagner. 2020. Stateful detection of black-box adversarial attacks. In *Proceedings of the 1st ACM Workshop on Security and Privacy on Artificial Intelligence*. ACM, 30–39.

[7] Si Chen, Mostafa Kahla, Ruoxi Jia, and Guo-Jun Qi. 2021. Knowledge-Enriched Distributional Model Inversion Attacks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 16178–16187.

[8] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. 2017. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526* (2017).

[9] Christopher A Choquette-Choo, Florian Tramer, Nicholas Carlini, and Nicolas Papernot. 2021. Label-only membership inference attacks. In *International Conference on Machine Learning*. PMLR, 1964–1974.

[10] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. 2004. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*. 253–262.

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.

[12] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. 2015. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. 1322–1333.

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

[14] David Gollob. 2015. *Microsoft Azure-Planning, Deploying, and Managing Your Data Center in the*. Springer-verlag Berlin And Hei.

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

[16] Bo Hui, Yuchen Yang, Haolin Yuan, Philippe Burlina, Neil Zhenqiang Gong, and Yinzhi Cao. 2021. Practical blind membership inference attack via differential comparisons. *arXiv preprint arXiv:2101.01341* (2021).

[17] Jinyuan Jia, Ahmed Salem, Michael Backes, Yang Zhang, and Neil Zhenqiang Gong. 2019. Memguard: Defending against black-box membership inference attacks via adversarial examples. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*. 259–274.

[18] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data* 7, 3 (2019), 535–547.

[19] Mostafa Kahla, Si Chen, Hoang Anh Just, and Ruoxi Jia. 2022. Label-Only Model Inversion Attacks via Boundary Repulsion. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, USA, 15025–15033. https://doi.org/10.1109/CVPR52688.2022.01462

[20] Tero Karras, Samuli Laine, and Timo Aila. 2019. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 4401–4410.

[21] Daniel Kifer, Adam Smith, and Abhradeep Thakurta. 2012. Private convex empirical risk minimization and high-dimensional regression. In *Conference on Learning Theory*. JMLR Workshop and Conference Proceedings, 25–1.

[22] Konstantina Kourou, Themis P Exarchos, Konstantinos P Exarchos, Michalis V Karamouzis, and Dimitrios I Fotiadis. 2015. Machine learning applications in cancer prognosis and prediction. *Computational and structural biotechnology journal* 13 (2015), 8–17.

[23] Alex Krizhevsky and Geoffrey Hinton. 2009. *Learning multiple layers of features from tiny images*. Technical Report 0. University of Toronto, Toronto, Ontario.

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems* 25 (2012).

[25] Cassidy Laidlaw, Sahil Singla, and Soheil Feizi. 2020. Perceptual adversarial robustness: Defense against unseen threat models. *arXiv preprint arXiv:2006.12655* (2020).

[26] Amazon Machine Learning. 2018. Developer guide. *Amazon Web Services* (2018).

[27] Huiying Li, Shawn Shan, Emily Wenger, Jiayun Zhang, Haitao Zheng, and Ben Y Zhao. 2020. Blacklight: Defending black-box adversarial attacks on deep neural networks. *arXiv preprint arXiv:2006.14042* (2020).

[28] Huichen Li, Xiaojun Xu, Xiaolu Zhang, Shuang Yang, and Bo Li. 2020. Qeba: Query-efficient boundary-based blackbox attack. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 1221–1230.

[29] Jiacheng Li and Bruno Ribeiro Ninghui Li. 2020. Membership Inference Attacks and Defenses in Supervised Learning via Generalization Gap. *arXiv* (2020).

[30] Zheng Li and Yang Zhang. 2021. Membership leakage in label-only exposures. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 880–895.

[31] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. 2015. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*. 3730–3738.

[32] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. 2015. Deep Learning Face Attributes in the Wild. In *Proceedings of International Conference on Computer Vision (ICCV)*.

[33] Yunhui Long, Vincent Bindschaedler, and Carl A Gunter. 2017. Towards measuring membership privacy. *arXiv preprint arXiv:1712.09136* (2017).

[34] Milad Nasr, Reza Shokri, and Amir Houmansadr. 2018. Machine learning with membership privacy using adversarial regularization. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*. 634–646.

[35] Hong-Wei Ng and Stefan Winkler. 2014. A data-driven approach to cleaning large face datasets. In *2014 IEEE international conference on image processing (ICIP)*. IEEE, 343–347.

[36] Ahmed Salem, Apratim Bhattacharya, Michael Backes, Mario Fritz, and Yang Zhang. 2020. {Updates-Leak}: Data Set Inference and Reconstruction Attacks in Online Learning. In *29th USENIX security symposium (USENIX Security 20)*. 1291–1308.

[37] Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. 2018. Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. *arXiv preprint arXiv:1806.01246* (2018).

[38] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*. IEEE, 3–18.

[39] Liwei Song, Reza Shokri, and Prateek Mittal. 2019. Privacy risks of securing machine learning models against adversarial examples. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 241–257.

[40] Shuang Song, Kamalika Chaudhuri, and Anand D Sarwate. 2013. Stochastic gradient descent with differentially private updates. In *2013 IEEE Global Conference on Signal and Information Processing*. IEEE, 245–248.

[41] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. 2012. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402* (2012).

[42] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. 2012. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks* 32, 0 (2012), –. https://doi.org/10.1016/j.neunet.2012.02.016

[43] Di Wang, Minwei Ye, and Jinhui Xu. 2017. Differentially private empirical risk minimization revisited: Faster and more general. *Advances in Neural Information Processing Systems* 30 (2017).

[44] Tianhao Wang, Yuheng Zhang, and Ruoxi Jia. 2021. Improving robustness to model inversion attacks via mutual information regularization. *Proceedings of the AAAI Conference on Artificial Intelligence* 35, 13 (2021), 11666–11673.

[45] Lior Wolf, Tal Hassner, and Itay Maoz. 2011. Face recognition in unconstrained videos with matched background similarity. In *CVPR 2011*. IEEE, 529–534.

[46] Ziqi Yang, Bin Shao, Bohan Xuan, Ee-Chien Chang, and Fan Zhang. 2020. Defending model inversion and membership inference attacks via prediction purification. *arXiv preprint arXiv:2005.03915* (2020).

[47] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. 2018. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 586–595.

[48] Yuheng Zhang, Ruoxi Jia, Hengzhi Pei, Wenxiao Wang, Bo Li, and Dawn Song. 2020. The secret revealer: Generative model-inversion attacks against deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 253–261.

A PROOF OF PROPOSITION 4.2

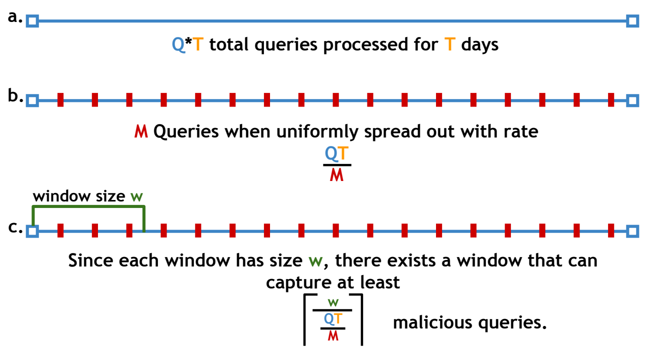

a.
Q*T total queries processed for T days
b.
M Queries when uniformly spread out with rate
QT/M
window size w
c.
Since each window has size w, there exists a window that can capture at least
⌈w / (QT/M)⌉ malicious queries.

Figure 4: M Malicious queries within T processing days with a window size w.

Proof. Let S be a detection system with a sliding window of size w that processes Q number of queries per day. Consider an attacker which requires at least $\mathcal{M}$ number of malicious queries for a successful deployment of an attack within T days. First, let's observe that the number of total queries processed by S in T days equals $Q \cdot T$ (Fig. 4 a.). Then if the $\mathcal{M}$ malicious queries are uniformly placed, then they occur with a rate of $\frac{Q \cdot T}{M}$ (Fig. 4 b.). Since our windows size is w, then by the pigeonhole principle, we can guarantee that there exists a window that has at least $\left\lceil \frac{w}{\left\lceil \frac{Q \cdot T}{\mathcal{M}} \right\rceil} \right\rceil$ number of malicious queries (Fig. 4 c.). **Q.E.D.**

B ADDITIONAL INFORMATION AND RESULTS FOR SECTION 5

B.1 Query Budgets for Privacy Attacks

We provide the number of malicious queries that an adversary has to send to complete the various privacy attacks in Table 12. Except for DA, the attacker has to send thousands of queries to finish HSJA or QEBA for one target point. Moreover, the attacker needs more than 55,000 queries to successfully reconstruct one target attribute. We propose to use the sliding window technique to improve efficiency, since if we have a reasonable window size, we can still mitigate the attack performance, considering the large amount of query budget.

We further present the number of queries PrivMon accepts to detect a malicious query in Table 11. As described in the table, our system only accepts at most 2 malicious queries to detect attacks out of $4 - 5k$ (HSJA/QEBA), and $50 - 60k$(BREPMI) queries.

B.2 Comparison between PrivMon and Feature KNN

To further investigate the system overhead of PrivMon and Feature KNN, we draw line charts for the CPU usage, memory usage, and average processing time of a single query as the number of queries increases. The results are shown in Figure 5. From these figures, we can observe that PrivMon has better performance than Feature KNN for all these three metrics. That is because 1) Feature KNN uses the traditional KNN algorithm, which is more computation-intensive; 2) Feature KNN stores all previous queries; 3) the computation complexity of KNN is linear to the number of samples.

We summarize the results for detection and mitigation performance in Table 6. As presented in the table, Feature KNN shows the comparable (e.g., HSJA, QEBA, DA ($d1$), MIAs) or slightly superior performance (e.g., (DA ($r5$))), in comparison to PrivMon. The reason behind this is Feature KNN retains all previous queries, which might lead to an increase in performance. However, it is important to mention that Feature KNN has a significant limitation in terms of efficiency since it stores all queries, which limits its applicability in real-time systems.

B.3 DSR Comparison under DA with Parallel Attacks

We continue our analysis of the DSR while taking into account the parallel attack scenario. In this experiment, we employ 1K data points while maintaining the same window size $w = 5k$, since we present the experimental result with 2,500 data points in Table 4. In this case, the defender only receives $2 - 3$ queries that are semantically related to one another within one window. According to the results presented in Table 10, our system outperforms Input LSH and Feature KNN by a margin of 0.17 to 0.28 for translation and 0.06 to 0.26 for rotation.

C ABLATION STUDY FOR UNIVERSAL THRESHOLD SELECTION

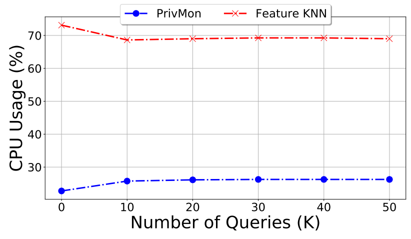
We further explain how to select the universal threshold for all our evaluations. We note that we use the same universal threshold over all the experiments. We vary the threshold from 250 to 450 to find an appropriate threshold. As shown in Figure 6, we consider two factors: attack mitigation performance and false positive rate, since the effective threshold should guarantee a high attack mitigation performance with a low false positive rate. When we set the threshold as 350, we can achieve the maximum attack mitigation performance (BREPMIs nearly 10% accuracy, DA[MFA] nearly 0.500 AUC score, and HSJA[MFA] near 0.500 AUC score), while preserving the reasonable False Positive Rate (<0.1). [DA: Data Augmentation/ HSJA: HopSkipJumpAttack/ BREPMI: Model Inversion Attack/ MFA: Membership inFerence Attack]

D ABLATION STUDY FOR SYSTEM WINDOW SIZE

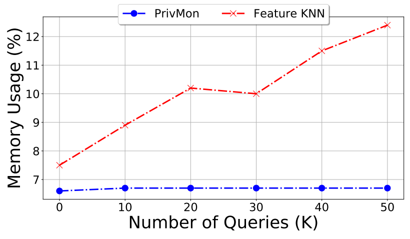
We compare the system performance against sequential DA attacks for window sizes varying from 5K to 105K. We evaluated the system performance (AUC score) and the system overhead (memory usage, CPU usage on a single core, and throughput), shown in Table 13. The results show that there is no significant increase in the system overhead as the window size becomes larger. Also, we observe a slight drop in system performance as the window size exceeds 45K. This is because a large window size increases the possibility of similar benign queries being included in the sliding window, thereby increasing the false positive rate. We conclude that a window size

Table 6: Detection and mitigation performance comparison between PrivMon and Feature KNN.

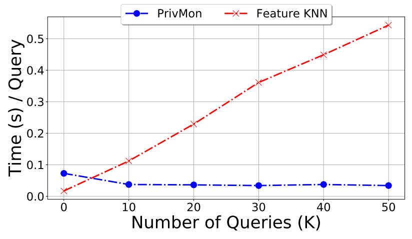
Attack Type		Dataset	Detection			w/o	Mitigation				
			Random	Feature KNN	PrivMon	Defense	Random	Feature KNN		PrivMon	
			DSR	DSR	DSR	ASR	ASR	ASR	FPR	ASR	FPR
MFAs	HSJA	CIFAR10	0.500	1.000	1.000	0.852	0.500	0.500	0.000	0.500	0.000
		CIFAR100	0.500	1.000	1.000	0.955	0.500	0.500	0.000	0.500	0.000
	QEBA	CIFAR10	0.500	1.000	1.000	0.917	0.500	0.500	0.000	0.500	0.000
		CIFAR100	0.500	1.000	1.000	0.966	0.500	0.500	0.000	0.509	0.000
	DA ($d1$)	CIFAR10	0.500	0.884	0.882	0.793	0.500	0.509	0.053	0.535	0.019
		CIFAR100	0.500	0.878	0.879	0.916	0.500	0.718	0.086	0.739	0.045
	DA ($r5$)	CIFAR10	0.500	0.959	0.948	0.808	0.500	0.605	0.101	0.500	0.016
		CIFAR100	0.500	0.957	0.951	0.938	0.500	0.718	0.086	0.573	0.038
MIAs	BREPMI	FaceScrub	0.500	0.695	0.697	0.650	0.000	0.110	0.000	0.100	0.001
		CelebA	0.500	0.923	0.926	0.890	0.000	0.130	0.000	0.120	0.000



(a) CPU Usage

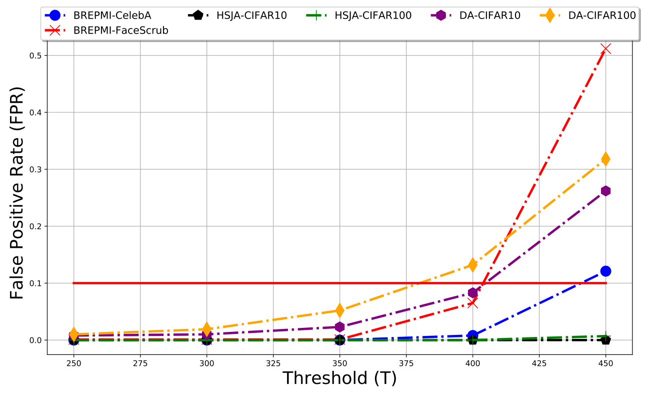


(b) Memory Usage



(c) Processing Time

Figure 5: System overhead and efficiency comparison between PrivMon and Feature KNN.



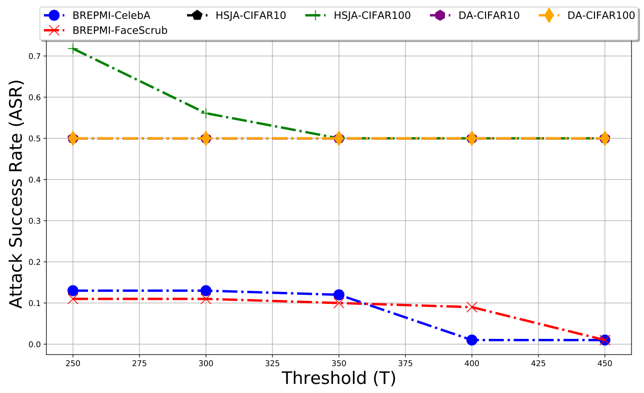


Figure 6: The False Positive Rate and Attack Success Rate for various privacy attacks on different datasets with the threshold change. These two metrics indicate the system performance according to the corresponding threshold, so we use those metrics to choose the universal threshold.

Table 7: Dataset Information for Membership Inference Attacks.

Dataset	# Classes	# Training data	# Test data	Input size
CIFAR10 [23]	10	50,000	10,000	32 × 32
CIFAR100 [23]	100	50,000	10,000	32 × 32
GTSRB [42]	43	39,209	12,630	64 × 64

between 5K and 45K would be a good choice on the trade-off between efficiency and performance. We apply a 5K window size for the rest of our experiments for efficiency.

Table 8: Dataset Information for Model Inversion Attacks.

Dataset	# Images	# Total id	# Public id	# Private id	# Target id
CelebA [32]	202,599	10,177	9,177	1,000	100
FaceScrub [35]	106,863	530	330	200	100

E EXPERIMENT SETUP ON VIDEO DATA

When we evaluate Table 3, we use two video datasets: YouTubeFaces [45] and UCF101 [41].YouTubeFaces dataset consists of 3,425 videos of 1,595 identities and has been used for face recognition.

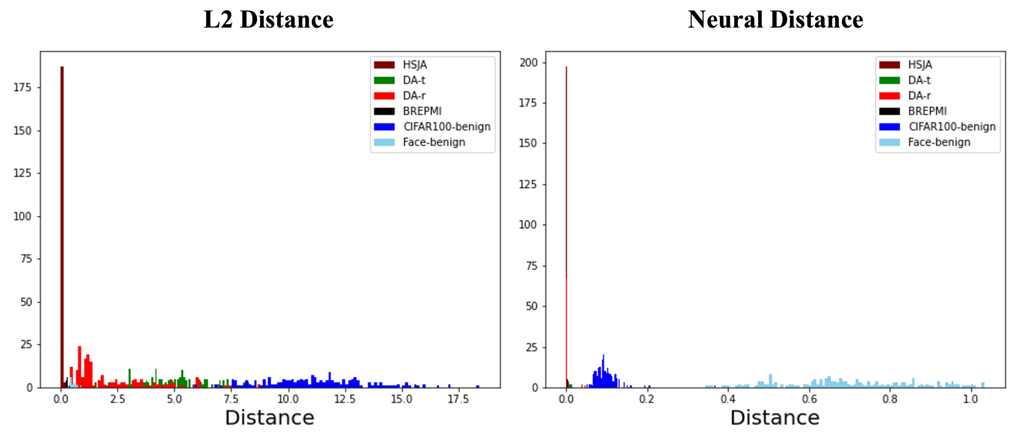


Figure 7: Distances to the nearest neighbor. As shown in the figure, l_2 distance makes malicious and benign queries entangled, so it is difficult to correctly identify malicious queries. On the other hand, *Neural distance* provide a clear separation between those queries to classify malicious one correctly.

Table 9: The attack mitigation performance of Feature KNN with $K = 50$ on DA. We present the attack success rate (ASR) and the false positive rate (FPR).

K=50	Dataset	Feature KNN	
		ASR	FPR
DA ($r5$)	CIFAR10	0.500	0.474
	CIFAR100	0.500	0.494
DA ($d1$)	CIFAR10	0.500	0.444
	CIFAR100	0.500	0.464

Table 10: The DSR of different approaches in detecting malicious queries of (1K) parallel DA attacks.

Attack Type	Dataset	PrivMon	Input LSH	Feature KNN
DA ($d1$)	CIFAR10	0.84	0.67	0.56
	CIFAR100	0.84	0.67	0.56
DA ($r5$)	CIFAR10	0.89	0.83	0.63
	CIFAR100	0.89	0.82	0.63

Table 11: The number of queries our system has to accept to first detect the malicious query. Our system only accept 1 malicious query and can detect the second malicious query for iteration-based privacy attacks.

Attack Type		Dataset	Queries to Detect
MFA	HSJA	CIFAR10	2
		CIFAR100	2
	QEBA	CIFAR10	2
		CIFAR100	2
MIA	BREPMI	FaceScrub	258 [2]
		CelebA	256 [2]

We extract video frames and crop the frames based on the work [8]. After this, we choose classes with more than two sub-classes to split the data into the train and validation datasets. Then, we randomly split the train data and validation data into the data for the original purposes (i.e., training and validation) and the data for attack (i.e., augmentation). To reduce the model complexity, we randomly choose 50 classes out of the total number of classes.

UCF101 dataset contains 13,320 videos from 101 action categories and has been exploited for human action recognition. We use the entire class and split the dataset into train and validation sets. We split each dataset into the data for model training and validation and the data for the attack. Here, we sample the images that are semantically similar to each other. Note that the images that belong to the same class from YouTubeFaces datasets are intrinsically similar because they indicate the same identity. However, UCF101 is designed for action recognition, so it contains semantically dissimilar images. Therefore, we need to sample the images with similar semantic information to demonstrate the attack success rate.

For calculating the detection success rate (DSR), we need to guarantee that there is no overlap among benign queries. However, due to the nature of video datasets, it is difficult to sample a large number of entirely distinct benign queries. Therefore, we consider 200 benign and 200 malicious samples in our experiments to calculate the DSR. This metric will provide the system performance for detecting semantically similar queries.

In addition, model 1 (i.e., $\mathcal{M}1$) consists of four convolutional layers with two fully connected layers. We further consider the complex model; model 2 (i.e., $\mathcal{M}2$) is composed of twelve convolutional layers with two fully connected layers.

F DATASET DESCRIPTION

We provide details about the datasets we evaluated. We make use of the CIFAR10 and CIFAR100 [23], which consist of 10 classes and 100 classes with 60K images, respectively. The resolution for those datasets is 32x32. Additionally, we take into account the GTSRB dataset [42], which is composed of 43 classes with 64x64 resolution.

The aforementioned datasets do not contain the sensitive information to reconstruct for model inversion attacks. Therefore, existing works have focused on reconstructing face images, such

Table 12: The number of queries that an adversary has to send for various privacy attacks to complete the attacks. We take an average over the number of target points to calculate the average number of queries. DA ($r5$) denotes rotation-based MFA and DA ($d1$) indicates translation-based MFA.

Attack Type	**HSJA**		**QEBA**		**BREPMI**	**BREPMI**	**DA ($d1$)**	**DA ($r5$)**
Dataset	**CIFAR10**	**CIFAR100**	**CIFAR10**	**CIFAR100**	**CelebA**	**FaceScrub**	**CIFAR10, CIFAR100**	**CIFAR10, CIFAR100**
Avg # of Queries	4232.76	3954.23	5040.1	4714.79	56780.29	57778.8	5	11
# of Target Samples	100	100	100	100	100	100	1000	1000

Table 13: Ablation study for the sliding window size of the stream-based system. For window size varies from 5K to 105K, we evaluate the AUC score, memory usage (MB), CPU usage on a single core (%), and throughput (Millions/Day).

Window Size	**Attack Type**															
	CIFAR10 ($d1$)				**CIFAR10 ($r5$)**				**CIFAR100 ($d1$)**				**CIFAR100 ($r5$)**			
	AUC	**Mem**	**CPU**	**Thr**	**AUC**	**Mem**	**CPU**	**Thr**	**AUC**	**Mem**	**CPU**	**Thr**	**AUC**	**Mem**	**CPU**	**Thr**
5K	0.929	2530	72.3	2.62	0.951	2530	77.1	2.63	0.881	2530	72.8	2.62	0.880	2530	77.0	2.63
25K	0.881	2530	72.8	2.62	0.948	2530	78.4	2.63	0.951	2530	72.8	2.63	0.950	2530	77.3	2.63
45K	0.877	2530	72.5	2.63	0.948	2530	77.6	2.63	0.948	2530	72.9	2.63	0.947	2530	78.1	2.62
65K	0.886	2530	73.1	2.63	0.947	2530	78.4	2.63	0.889	2530	73.4	2.63	0.951	2530	77.5	2.62
85K	0.887	2530	73.1	2.63	0.949	2530	78.0	2.63	0.885	2530	73.6	2.63	0.948	2530	78.9	2.63
105K	0.879	2530	73.5	2.64	0.936	2530	78.9	2.63	0.874	2530	73.6	2.64	0.950	2530	78.8	2.63

as CelebA [32] consisting of 10,177 identities with over 200K images and FaceScrub [35] which includes 530 identities with over 100K images, given an identity [7, 19, 48]. They leverage the public data [20] to train GANs to reach the meaningful optimal point. After dividing the total identities into public and private classes, the public classes were used to train the GAN (which we will train the target model). There are no identities that overlap between the public and private domains. This indicates that the attacker is completely unaware of the identities within the private domain. Then, we execute the attack against the classifier trained on the private domain. The detailed attack setting is described in [19].

G COMPARISON BETWEEN l_2 AND NEURAL DISTANCE

We provide the comparison results between l_2 distance and neural distance on different types of attacks (DA ($d1$), DA ($r5$), HSJA, BREPMI). We sample 200 benign (from CIFAR100, and CelebA), and malicious queries (from each attack), then calculate the distance to the nearest neighbor. As described in Figure 7, if we leverage the neural distance, we can find a separation between malicious queries and benign queries. However, if we rely on l_2 distance, we cannot distinguish those queries. This indicates that neural distance can provide a reliable nearest-neighbor search performance.