Searching Privately by Imperceptible Lying: A Novel Private Hashing Method with Differential Privacy

Yimu Wang, Shiyin Lu, and Lijun Zhang
{wangym,lusy,zhanglj}@lamda.nju.edu.cn
State Key Laboratory for Novel Software Technology, Nanjing University
Nanjing 210023, China

ABSTRACT

In the big data era, with the increasing amount of multi-media data, approximate nearest neighbor (ANN) search has been an important but challenging problem. As a widely applied large-scale ANN search method, hashing has made great progress, and achieved sub-linear search time with low memory space. However, the advances in hashing are based on the availability of large and representative datasets, which often contain sensitive information. Typically, the privacy of this individually sensitive information is compromised. In this paper, we tackle this valuable yet challenging problem and formulate a task termed as *private hashing*, which takes into account both searching performance and privacy protection. Specifically, we propose a novel noise mechanism, *i.e.*, Random Flipping, and two private hashing algorithms, *i.e.*, PHashing and PITQ, with the refined analysis within the framework of differential privacy, since differential privacy is a well-established technique to measure the privacy leakage of an algorithm. Random Flipping targets binary scenarios and leverages the "Imperceptible Lying" idea to guarantee ϵ-differential privacy by flipping each datum of the binary matrix (noise addition). To preserve ϵ-differential privacy, PHashing perturbs and adds noise to the hash codes learned by non-private hashing algorithms using Random Flipping. However, the noise addition for privacy in PHashing will cause severe performance drops. To alleviate this problem, PITQ leverages the power of alternative learning to distribute the noise generated by Random Flipping into each iteration while preserving ϵ-differential privacy. Furthermore, to empirically evaluate our algorithms, we conduct comprehensive experiments on the image search task and demonstrate that proposed algorithms achieve equal performance compared with non-private hashing methods.

CCS CONCEPTS

• **Security and privacy**; • **Information systems** → **Information retrieval**;

KEYWORDS

Differential Privacy, Large-Scale Multimedia Retrieval, Hashing

ACM Reference Format:
Yimu Wang, Shiyin Lu, and Lijun Zhang. 2020. Searching Privately by Imperceptible Lying: A Novel Private Hashing Method with Differential Privacy. In *Proceedings of the 28th ACM International Conference on Multimedia (MM '20), October 12–16, 2020, Seattle, WA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3394171.3413882

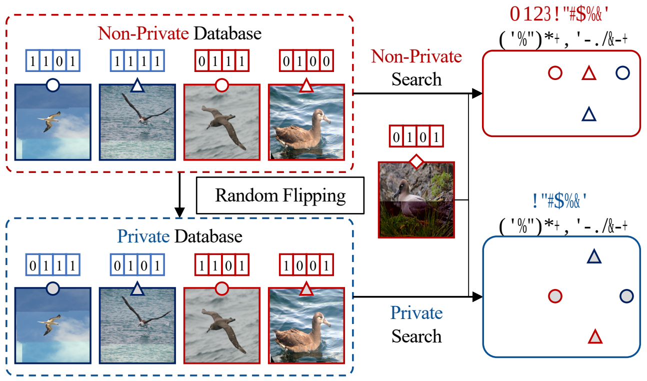


Figure 1: Our proposed *Random Flipping* is the first noise mechanism aimed at hashing scenarios, which changes the *non-private* database into a *private* database, and preserves ϵ-differential privacy. With non-private databases, privacy is likely to leak while attaining high search performance. With private databases, privacy will be protected but protecting privacy will cause a slight performance drop. Here, different colors of markers and boxes outside images represent different subordinate categories. d is the Hamming distance.

1 INTRODUCTION

To deal with the large-scale data, as a popular technique in multimedia search, approximate nearest neighbor (ANN) search [3, 4] has attracted much attention in recent years. Among many ANN search methods, hashing [17, 33, 35, 43, 51] is an active and representative sub-area. Particularly, hashing maps data points to binary codes with learned hash functions by preserving similarity in the original space of data points. With the power of the binary representation, the storage cost for the large-scale database can be drastically reduced, while the search time complexity is constant or sub-linear. Existing hashing methods can be roughly categorized into unsupervised and supervised hashing. Unsupervised hashing methods, *e.g.*, LSH [17], SH [51], and ITQ [18], learn hash functions from unlabeled data. To improve the quality of hash codes, supervised hashing methods such as KSH [33], utilize supervised information (*e.g.*, similarity matrix or label information). Inspired by the progress of deep neural networks [21, 50], deep supervised hashing [5, 20, 24, 30, 46, 53, 54] has been proposed, which leverages the power of deep learning to generate high-level semantic features.

With the increasing massive data gathered from customers or individuals in the real world, hashing has been applied in many scenarios, such as person re-identification [56] and face search [31], and achieved promising performance. The availability of large and representative datasets [26, 29], which are often crowd-sourced and contain sensitive information, advances the success of hashing technique. However, that would comprise and leak the privacy of individually sensitive information. The wide use of these sensitive data requires techniques to meet the demands of applications that offer principled and rigorous privacy guarantees. Differential privacy [13] is a well-established metric that mathematically requires the probability of any observable outputs to change very little when any single datum changes. It ensures no attacker can tell whether the algorithm utilizes or not a specific training example, hence addressing the aforementioned problem. Recently, differential privacy has been applied in many active areas, *e.g.*, online learning [2, 19, 22], generic deep learning [1, 44], and natural language processing [37].

In this paper, we formulate this novel problem called *private hashing* as a step towards efficient large-scale multi-media search with privacy. To achieve private hashing, we propose a noise mechanism, Random Flipping, and two algorithms, *i.e.*, PHashing and PITQ, within the framework of differential privacy.

The proposed noise mechanism, Random Flipping, leverages the power of uncertainty ("Imperceptible Lying" proposal) to protect privacy. Consider the following scenario: We are taking a test questioning some private individual information, and we have to answer "yes" or "no". If we want to keep this private answer uncovered, we can simply protect our privacy by randomly changing our reply. Similarly, the binary hash codes can be regarded as sequential questions of input features, such as black bird head and yellow long tail, which can be used for identifying the input. Random Flipping is based on this proposal and preserves ϵ-differential privacy by randomly flipping each datum (private question) of binary input with a predefined probability as shown in Fig. 1.

Two private hashing algorithms target different scenarios. PHashing utilizes Random Flipping to extend non-private hashing methods into private methods and preserves ϵ-differential privacy. However, the privacy protection causes severe search performance drops, which are inversely proportional to the strength of privacy-preserving. To alleviate this problem, we develop PITQ, which leverages the power of iterative discrete optimization to distribute the noise generated by Random Flipping into several iterations with ϵ-differential privacy.

To empirically evaluate the search performance and cost of privacy protection of our proposed algorithms, *i.e.*, PHashing and PITQ, we conduct comprehensive experiments on five vision datasets, *i.e.*, *ImageNet* [11], *CIFAR-10* [26], *NUS-WIDE* [10], *Aircraft* [36], and *CUB* [48]. Empirical results show that PHashing preserves ϵ-differential privacy at an acceptable price. PITQ achieves equal performance compared with non-private ITQ, and sometimes outperforms the non-private ITQ as the noise helps to converge to a better local optimum in the discrete binary optimization.

The main contributions of our paper are summarized as follows:

- Our paper is the first to formulate the practical and challenging *private hashing* problem within the framework of differential privacy.
- The proposed *Random Flipping* is the first noise mechanism aimed at hashing scenarios in the differential privacy area, which ensures ϵ-differential privacy by flipping each datum of the binary input with a predefined probability.
- We devise two private hashing algorithms preserving ϵ-differential privacy, *i.e.*, PHashing and PITQ, and present theoretical guarantees. PHashing extends the non-private hashing algorithms into private algorithms by Random Flipping with an acceptable price. To reduce the performance drop, PITQ distributes the noise generated by Random Flipping into several iterations instead of adding the noise only in the last iteration, which leads to better performances than PHashing (ITQ).
- Empirical results show that our algorithms achieve equal performances compared with non-private algorithms on the image search task while preserving ϵ-differential privacy.

2 RELATED WORK

In this section, we briefly review related work on differential privacy and hashing.

2.1 Differential Privacy

Differential privacy, firstly introduced by Dwork et al. [13], is a mathematical notion of privacy that requires the probability of any observable outputs to change very little when any single datum changes.

Definition 1 (Differential Privacy [14]). *A randomized algorithm* $\mathcal{M}$ *with domain* $\mathbb{N}^{|\mathcal{X}|}$ *is* (ϵ,δ)*-differentially private if* $\forall S \subseteq \mathrm{Range}(\mathcal{M})$ *and* $\forall x, y \in \mathbb{N}^{|\mathcal{X}|}$ *with* $\|x-y\|_1 \le 1$,

$$\Pr[\mathcal{M}(x) \in S] \le e^{\epsilon} \Pr[\mathcal{M}(y) \in S] + \delta . \qquad (1)$$

Here, $\epsilon, \delta \ge 0$ *are privacy loss parameters, which quantify the strength of privacy protection. If* S *is a countable set, then we can modify the inequality* (1) *as*

$$\Pr[\mathcal{M}(x) = s] \le e^{\epsilon} \Pr[\mathcal{M}(y) = s] + \delta , \qquad (2)$$

where $s \in S$*. If* $\delta = 0$*, we say that* $\mathcal{M}$

randomized mapping. Then $f \circ \mathcal{M} : \mathbb{N}^{\|\mathcal{X}\|} \to \mathbb{R}'$ is (ϵ, δ)-differentially private.

In another word, if any part of an algorithm, including outputs and any components, is differentially private, the whole algorithm is differentially private. Another important property related to our work is the basic composition theorem, which indicates that the composition of k (ϵ_k, δ_k)-differential private algorithms is at worst $(\sum_k \epsilon_k, \sum_k \delta_k)$-differential privacy.

Theorem 3 (Basic Composition [14]). *Let $\mathcal{M}_i : \mathbb{N}^{\|\mathcal{X}\|} \to \mathbb{R}_i$ be (ϵ_i, δ_i)-differential private for $i = 1, \ldots, k$. Then the composition $\mathcal{M}_{[k]} : \mathbb{N}^{\|\mathcal{X}\|} \to \mathbb{R}_1 \times \cdots \times \mathbb{R}_k$ defined as: $\mathcal{M}_{[k]}(x) = (\mathcal{M}_1(x), \mathcal{M}_2(x), \ldots, \mathcal{M}_k(x))$ is $\left(\sum_{i=1}^k \epsilon_i, \sum_{i=1}^k \delta_i\right)$-differential private.*

A typical paradigm for achieving differentially private real-valued algorithms $\mathcal{M}_p$ is adding noise into the non-private algorithm $\mathcal{M}$. The noise is calibrated to the sensitivity of $\mathcal{M}$, which is the maximum of the absolute distance $\|\mathcal{M}(\mathrm{x}) - \mathcal{M}(\mathrm{x}')\|$, where x and x' are adjacent inputs, *i.e.*, $\|\mathrm{x} - \mathrm{x}'\|_1 \leq 1$. For instance, the Gaussian noise mechanism is defined by

$$\mathcal{M}_p(\mathrm{x}) = \mathcal{M}(\mathrm{x}) + \mathcal{N}\left(0, \max_{\mathrm{x},\mathrm{x}' \in \mathcal{X}, \|\mathrm{x}-\mathrm{x}'\|_1 \leq 1} \left(\|\mathcal{M}(\mathrm{x}) - \mathcal{M}(\mathrm{x}')\|\right)^2 \sigma^2\right), \tag{3}$$

where $\mathcal{N}\left(0, \max_{\mathrm{x},\mathrm{x}' \in \mathcal{X}, \|\mathrm{x}-\mathrm{x}'\|_1 \leq 1} \left(\|\mathcal{M}(\mathrm{x}) - \mathcal{M}(\mathrm{x}')\|\right)^2 \sigma^2\right)$ is the Gaussian distribution with mean 0 and standard deviation $\max_{\mathrm{x},\mathrm{x}' \in \mathcal{X}, \|\mathrm{x}-\mathrm{x}'\|_1 \leq 1} \left(\|\mathcal{M}(\mathrm{x}) - \mathcal{M}(\mathrm{x}')\|\right)\sigma$. If $\delta \geq \frac{4}{5}\exp^{-(\delta\epsilon)^2/2}$ and $\epsilon < 1$, the private real-valued algorithm $\mathcal{M}_p$ satisfies (ϵ, δ)-differential privacy [14]. This paradigm utilizes the post-processing theorem by implying a real-valued random Gaussian noise.

However, the existing literature on differential privacy mostly targets real-valued algorithms, *e.g.*, boosting [15], principal component analysis [9], linear and logistic regression [7, 55], support vector machines [38], risk minimization [8, 12], continuous data processing [39], online learning [2, 19, 22], generic deep learning [1, 44] and natural language processing [37], as opposed to *binary algorithms*, such as hashing [20]. As the privacy of training data will be leaked in the hashing scenarios, hashing is facing more critical situations, and there does not exist suitable solutions for hashing. We firstly formulate this practical and challenging *private hashing* problem, and develop Random Flipping for the binary situation.

2.2 Hashing

Hashing is a widely used method for ANN search in large scale multi-media search with encouraging efficiency in both speed and storage. Existing hashing methods can be roughly categorized into unsupervised and supervised hashing. Unsupervised hashing methods [25] learn hash functions from unlabeled data, *e.g.*, LSH [17], SH [51] and ITQ [18]. Locality Sensitive Hashing (LSH) [17] uses random projections as the hash function. Iterative quantization (ITQ) alternatively rotates the projected data and updates the codes to find an optimal rotation for minimizing the quantization loss. Supervised hashing methods [23, 28, 34, 47, 49, 52] leverage supervised information (*e.g.*, similarity matrix or label information) to improve the quality of hash codes, *e.g.*, KSH [33] and SDH [42]. KSH [33] utilizes the property of kernels to simplify the original problem while achieving better search accuracy.

Inspired by powerful feature representations in learning with deep neural networks [27], deep supervised hashing [6, 16] that adopts deep learning to generate high-level semantic features has been proposed. Deep Pairwise-Supervised Hashing (DPSH) [30] preserves relative similarities between image triplets straightly by integrating feature learning and hash functions in an end-to-end manner. Deep Supervised Hashing (DSH) [32] splits training data into similar pairs and dissimilar pairs to generate similarity correlations and controls the quantization error to further improve the performance. Moreover, HashNet [5] tackles the data imbalance problem between similar and dissimilar pairs and alleviates this problem by adjusting the weights of similar pairs. To additionally accelerate the training procedure, several asymmetric deep hashing methods [41] are proposed, *e.g.*, Asymmetric Deep Supervised Hashing (ADSH) [24], which only learns the hash function for query points to reduce the computational complexity. For deriving compact hash codes, the objective functions in hashing learning are always designed by the following three principles: i) preserving the similarity of data points in the original space, ii) distributing the codes to uniformly fulfill the code space, and iii) generating compact binary codes.

While the aforementioned hashing methods have shown great success in multi-media searching, they ignore the privacy of data. In this paper, to protect privacy which will be potentially revealed, we are the first to investigate the *private hashing* problem and devise two private hashing algorithms with a private noise mechanism. Besides, to theoretically analyze the proposed algorithms, we derive the privacy and stability guarantees for measuring the influence of hash codes by noise addition.

3 METHOD

In this section, we elaborately introduce our noise mechanism for privacy protection and our proposed private hashing methods. Besides, we present theoretical guarantees of our algorithms.

Suppose that we have m query data points and n database points denoted as $\mathrm{X} = \{\mathrm{x}_i\}_{i=1}^m$ and $\mathrm{Y} = \{\mathrm{y}_j\}_{j=1}^n$. The pairwise information is denoted by $\mathrm{S} \in \{-1, +1\}^{m \times n}$, which is available in the supervised setting. If point x_i and point y_j are similar, $\mathrm{S}_{ij} = +1$, otherwise $\mathrm{S}_{ij} = -1$. Under this definition, the goal of hashing is to learn binary hash codes $\mathrm{b} \in \{-1, +1\}^c$ for each point, and the corresponding hash function $\mathrm{h}(\cdot)$, where c is the target length of binary codes. We denote $\mathrm{V} \in \{-1, +1\}^{m \times c}$ and $\mathrm{U} \in \{-1, +1\}^{n \times c}$ as the hash codes of the query points and the database points, respectively.

The mean average precision (mAP) is one of the most widely used criteria to evaluate search accuracy. The core idea of mAP is to evaluate ranked lists by averaging the precision at each position. Given m query $\mathrm{V} = [\mathrm{v}_1^\top, \ldots, \mathrm{v}_m^\top] \in \{-1, +1\}^{m \times c}$ and lists of n ranked retrieval database results, mAP is defined as:

$$mAP(\mathrm{V}) = \frac{1}{m}\sum_{i=1}^n \frac{1}{N_i^+}\sum_{j=1}^n \mathrm{precision}(\mathrm{v}_i, \mathrm{U}, \mathrm{S}, j)\mathbb{I}(\mathrm{S}_{ij} = 1)$$

where N_i^+ represents the count of ground-truth relevant instances in the database for the query v_i, $\mathbb{I}(\cdot)$ is the indicator function, and $\mathrm{precision}(\mathrm{v}_i, \mathrm{U}, \mathrm{s}, j)$ stands for the precision at cut-off j in the

Algorithm 1 RandomFlipping(U, β)

```
Input: $U \in \{-1, +1\}^{n \times c}$ represents the input binary matrix. $\beta$ is the hyper-parameter for controlling the strength of privacy-protecting.
Output: The perturbed binary matrix $U$.
1: for $i \to 1, \ldots, n$ do
2:     for $j \to 1, \ldots, c$ do
3:         Get a random number $0 \leq k < 1$;
4:         if $k < \beta$ then
5:             $U_{ij} \leftarrow -U_{ij}$;
6:         end if
7:     end for
8: end for
9: return $U$.
```

ranked list. mAP is related to the Hamming distance between query and database which is further influenced by hash codes of database.

3.1 Random Flipping

Here we present our novel noise mechanism, which is the key novelty of this paper and also an important component of the proposed private hashing algorithms.

The intuition behind our novel noise mechanism algorithm is "Imperceptible Lying". If we are taking a test questioning some private individual information and have to answer either "yes" or "no", and we want to keep this private answer uncovered, we can simply change our reply with a fixed probability. Similarly, "yes" and "no" can be seen as "+1" and "-1', while the c-length binary hash codes of an input data can be regarded as c sequential questions, such as "does the input data have black bird head?", which can be used for identifying the input. Hence, by changing each datum of c-length binary hash codes with a predefined probability β, the individually sensitive information of input data can be protected. This mechanism is suitable for the hashing scenario, in another word, the binary algorithms.

As shown in Algorithm 1, our proposed noise mechanism named as Random Flipping, takes a binary matrix, *e.g.*, hash codes U, as input, and randomly flips each datum of the input binary matrix with a predefined probability β, *i.e.*, flipping "+1" to "-1" or flipping "-1" to "+1", and then outputs the private binary matrix U'. Random Flipping is straightforward and has the following privacy guarantee:

Theorem 4 (Privacy Guarantee of Random Flipping). *Algorithm 1 is ϵ-differentially private, given $\beta = e^{-\epsilon}$, the parameter of Random Flipping.*

The above theorem shows that the size of the input matrix does not affect the privacy guarantee, while only the probability β will influence privacy protection. But, perturbing the input matrix will cause unpredictable consequences. To estimate the difference between the input matrix U and the perturbed matrix U' which preserves ϵ-differential privacy, we present the following lemma.

Lemma 5 (Stability of Input Binary Matrix). *Let $U \in \{-1, +1\}^{n \times c}$ be the input binary matrix and $U' \in \{-1, +1\}^{n \times c}$ be the perturbed binary matrix derived by $U' =$ RandomFlipping(U, β). The expected F-norm $\|U' - U\|_F$ satisfies*

$$\mathbb{E}\left[\left\|U' - U\right\|_F\right] = 2\sqrt{\beta nc}\,.$$

The above lemma can be further expanded as

$$\mathbb{E}\left[\left\|U' - U\right\|_F\right] = 2\sqrt{\beta nc} = 2\sqrt{\frac{nc}{e^{\epsilon}}}\,.$$

It indicates that the stability of the input matrix is proportional to the size of the input matrix and the privacy parameter ϵ, and inversely proportional to the strength of privacy-protecting, which is obvious in the sense if we want to hide more individual information, more stability will lose.

Algorithm 2 PHashing$(Y, S, H(\cdot, \cdot), \beta)$

```
Input: $Y = \{y_j\}_{j=1}^{n}$ represents $n$ data points. $S \in \{-1, +1\}^{n \times n}$ is the similarity matrix between data points $Y$. $H(\cdot, \cdot)$ is a hashing learning algorithm. $\beta$ is the hyper-parameter of the Random Flipping shown in Algorithm 1.
Output: $h(\cdot)$ is the learned hash function derived by $H(\cdot, \cdot)$. $U'$ is the private binary hash codes for data points.
1: Train: Learn the hash function $h(\cdot)$ and binary hash codes $U$ for data points by $h(\cdot), U \leftarrow H(Y, S)$.
2: Perturb: Obtain the private binary hash codes $U'$ for data points by $U' \leftarrow$ RandomFlipping$(U, \beta)$.
3: return $h(\cdot)$ and $U'$.
```

3.2 Private Hashing (PHashing)

In this part, we present our PHashing (private hashing) algorithm which can equip all the non-private hashing algorithms with the ability of protecting privacy.

Inspired by the post-processing theorem [14], PHashing combines our proposed novel *Random Flipping* noise mechanism with the non-private hashing algorithms to achieve private hashing. Specifically, after the training of a non-private hashing algorithm $H(\cdot, \cdot)$ and obtaining learned hash codes U and the hash function $h(\cdot)$, we perturb the learned hash codes U for privacy by RandomFlipping$(\cdot, \cdot)$. In this way, the perturbed hash codes U' is private. PHashing is widely applicable as we can employ all the existing hashing methods in the training procedure, and PHashing involves the following steps:

1. *Train:* Given any non-private hashing algorithm $H(\cdot, \cdot)$, we train it until convergence, and obtain the learned hash function $h(\cdot)$ and the binary hash codes U for the database;
2. *Perturb:* Given the privacy parameter β, we employ the Random Flipping mechanism shown in Algorithm 1 to derive the private hash codes U'.

The above procedure is summarized in Algorithm 2.

3.2.1 Privacy and Stability Analysis.

We now present the privacy and stability guarantees. With Theorem 2, we can easily obtain the following privacy guarantee.

Theorem 6 (Privacy Guarantee). *With $\beta = e^{-\epsilon}$, Algorithm 2 is ϵ-differentially private.*

Algorithm 3 PITQ(Y, β, E)

```
Input: $\mathrm{Y} = \{\mathrm{y}_j\}_{j=1}^{n}$ represents $n$ centralized data points. $\beta$ is the
    hyper-parameter of Random Flipping. $E$ is the iterative epoch.
Output: $\mathrm{U}$ and $\mathrm{R}$ are the private binary hash codes for data points
    and the rotation matrix.
1: Employ proper embedding methods, e.g., PCA, to project the
   database data $\mathrm{Y}$ and return the optimal projection matrix $\mathrm{W}$;
2: for $i \leftarrow 1, \ldots, E$ do
3:     Fix $\mathrm{R}$ and update $\mathrm{U}$: $\mathrm{U} \leftarrow \mathrm{sign}(\mathrm{YWR})$;
4:     Perturb $\mathrm{U}$: $\mathrm{U} \leftarrow \mathrm{RandomFlipping}(\mathrm{U}, \beta)$;
5:     Fix $\mathrm{U}$ and update $\mathrm{R}$: first compute the SVD of $\mathrm{U}^\top(\mathrm{YW})$ as
       $\mathrm{U}^\top(\mathrm{YW}) = \mathrm{N}\Omega\mathrm{N}^\top$, and then $\mathrm{R} \leftarrow \mathrm{NN}^\top$;
6: end for
7: return $\mathrm{R}$ and $\mathrm{U} = \mathrm{sign}(\mathrm{YWR})$.
```

Now, we focus on the stability of PHashing. Due to sparse and random properties of hash codes and the mechanism of calculating mAP, it is hard to measure the performance drop of an unconstrained algorithm, while if we add some assumptions about hashing algorithms and hash codes, the guarantee will not suit most of scenarios. Thus, for deriving general theorems estimating the privacy impact, we characterize the influence on the hash codes and Hamming distance between query and database, which directly influence the search accuracy, *i.e.*, mAP, and present the following lemma.

Lemma 7 (Stability of Hash Codes and Hamming Distance). *Let* $\mathrm{v} \in \{-1, +1\}^c$ *be a query code containing only one point. Let* $\mathrm{d}' \in \{0, \ldots, c\}^n$ *be the Hamming distance between* v *and the private hash code* $\mathrm{U}' \in \{-1, +1\}^{n\times c}$ *for the database, and* $\mathrm{d} \in \{0, \ldots, c\}^{n\times 1}$ *be the Hamming distance between* v *and the non-private hash code* $\mathrm{U} \in \{-1, +1\}^{n\times c}$*. The expected* F*-norm* $\left\|\mathrm{U} - \mathrm{U}'\right\|_F$ *satisfies Lemma 5, and expected* 1*-norm* $\left\|\mathrm{d} - \mathrm{d}'\right\|_1$ *satisfies*

$$\mathbb{E}\left[\left\|\mathrm{d} - \mathrm{d}'\right\|_1\right] \leq \frac{n^{3/2}c}{\sqrt{e^{\epsilon}}} = n^{3/2}c\sqrt{\beta}\,.$$

3.3 Private Iterative Quantization (PITQ)

In this section, to reduce performance drop, we devise a private unsupervised iterative hashing algorithm. The proposed algorithm is named as PITQ (private iterative quantization), whose performance is better than PHashing with $H(\cdot, \cdot) = \mathrm{ITQ}$. Different from PHashing, PITQ distributes the noise to each iteration and alternatively updates the interdependent parameters for improving the search performance, while PHashing adds the noise directly on the output.

As one of the most popular unsupervised conventional hashing methods, ITQ [18] first obtains the binary hash codes $\mathrm{U} = \mathrm{sign}(\mathrm{YWR})$ by rotating and quantizing the projected centralized data YW, where the projection matrix W is learned by embedding methods, *e.g.*, PCA, and then updates the rotation matrix R to minimize the quantization loss (Eq. (4)) iteratively. This procedure totally utilizes the structure of data points in the original space.

Inspired by the power of iterative discrete learning in ITQ, PITQ is motivated as follows: in each iteration $i \in [E]$, PITQ first obtains the binary hash codes $\mathrm{U} = \mathrm{sign}(\mathrm{YWR})$ by rotating (rotation matrix R) and quantizing the projected centralized data YW, where the projection matrix W is learned by embedding methods, *e.g.*, PCA. Now, for the sake of the privacy in the database, PITQ perturbs the learned hash codes U utilizing Random Flipping (Algorithm 1). After that, PITQ updates the rotation matrix R to minimize the quantization loss (Eq. (4)). In this way, the rotation matrix R and hash codes U are alternatively updated. In the last iteration, PITQ returns the learned rotation matrix R and private hash codes $\mathrm{U} = \mathrm{sign}(\mathrm{YWR})$, where the $\mathrm{sign}(\cdot)$ is the signum function. PITQ minimizes the following loss function:

$$Q(\mathrm{U}, \mathrm{R}) = \|\mathrm{U} - \mathrm{YWR}\|_F^2\,. \tag{4}$$

Specifically, PITQ involves the following steps:

1. *Fix* R *and update* U*:* Expanding Eq. (4), we have
$$\begin{aligned} Q(\mathrm{U}, \mathrm{R}) =& \|\mathrm{U}\|_F^2 + \|\mathrm{YW}\|_F^2 - 2\,\mathrm{tr}(\mathrm{UR}^\top(\mathrm{YW})^\top) \\ =& nc + \|\mathrm{YW}\|_F^2 - 2\,\mathrm{tr}(\mathrm{UR}^\top(\mathrm{YW})^\top)\,. \end{aligned} \tag{5}$$
Minimizing (5) is equivalent to maximizing $\mathrm{tr}(\mathrm{UR}^\top(\mathrm{YW})^\top)$. It is obvious that the trace is maximized by letting $\mathrm{U} = \mathrm{sign}(\mathrm{YWR})$;
2. *Perturb* U*:* Given the parameter β, we employ *Random Flipping* mechanism (Algorithm 1) to obtain private hash codes U;
3. *Fix* U *and update* R*:* Now the objective function (5) corresponds to the classic Orthogonal Procrustes problem [40], which tries to find a rotation for aligning two sets. Thus, we first compute the SVD of $\mathrm{U}^\top(\mathrm{YW})$ as $\mathrm{U}^\top(\mathrm{YW}) = \mathrm{N}\Omega\mathrm{N}^\top$, and then set $\mathrm{R} \leftarrow \mathrm{NN}^\top$;
4. Loop the step 1-3 for predefined iterations E, then return R and $\mathrm{U} = \mathrm{sign}(\mathrm{YWR})$.

As summarized in Algorithm 3, our PITQ algorithm alternately updates the parameters R and U for several predefined iterations E to find a locally optimal solution. The noise added when perturbing learned binary codes U in each iteration may help the discrete optimization to find a better local optimum in some cases practically, as shown in Tab. 2.

3.3.1 Privacy Analysis. Similarly, based on Theorem 2 and Theorem 3 of differential privacy, we have the following privacy guarantee.

Theorem 8 (Privacy Guarantee). *Algorithm 3 is* ϵ*-differentially private, given the parameter* β

Table 1: Comparisons of mAP w.r.t. the different lengths of bits on *ImageNet*, *CIFAR-10*, and *NUS-WIDE* of private versions (PHashing) and non-private versions (original algorithms) of several hashing algorithms. The rows ticked with private represents the private versions (PHashing) with ϵ-differential privacy.

Method	Private?	ϵ	ImageNet				CIFAR-10				NUS-WIDE			
			12 bits	24 bits	32 bits	48 bits	12 bits	24 bits	32 bits	48 bits	12 bits	24 bits	32 bits	48 bits
LSH		0	0.0520	0.1134	0.1625	0.2268	0.1162	0.1215	0.1264	0.1244	0.1850	0.2222	0.2088	0.2116
PHashing (LSH)	X	1	0.0130	0.0150	0.0162	0.0186	0.1036	0.1047	0.1057	0.1054	0.1704	0.1783	0.1752	0.1755
	X	2	0.0280	0.0532	0.0761	0.1140	0.1111	0.1147	0.1183	0.1170	0.1789	0.2052	0.1954	0.1968
	X	4	0.0475	0.1033	0.1492	0.2110	0.1154	0.1205	0.1252	0.1234	0.1841	0.2199	0.2069	0.2095
ITQ		0	0.2917	0.4929	0.5586	0.6337	0.1544	0.1607	0.1630	0.1656	0.3189	0.3264	0.3277	0.3322
PHashing (ITQ)	X	1	0.0213	0.0346	0.0451	0.0715	0.1110	0.1134	0.1148	0.1167	0.1960	0.1987	0.1999	0.2029
	X	2	0.1429	0.3192	0.3982	0.5116	0.1374	0.1442	0.1471	0.1509	0.2718	0.2805	0.2835	0.2907
	X	4	0.2684	0.4710	00.5389	0.6208	0.1522	0.1585	0.1612	0.1638	0.3128	0.3207	0.3223	0.3271
SH		0	0.1468	0.2545	0.3039	0.3739	0.1316	0.1289	0.1287	0.1274	0.2270	0.2134	0.2092	0.2066
PHashing (SH)	X	1	0.0142	0.0159	0.0168	0.0187	0.1061	0.1055	0.1054	0.1048	0.1724	0.1716	0.1716	0.1711
	X	2	0.0535	0.0917	0.1133	0.1591	0.1204	0.1185	0.1184	0.1171	0.1936	0.1890	0.1875	0.1862
	X	4	0.1275	0.2251	0.2712	0.3415	0.1299	0.1273	0.1272	0.1259	0.2205	0.2090	0.2050	0.2028
KSH		0	0.1458	0.1913	0.2150	0.2544	0.2339	0.2586	0.2669	0.2751	0.2891	0.2813	0.2773	0.3055
PHashing (KSH)	X	1	0.0148	0.0182	0.0206	0.0257	0.1228	0.1320	0.1387	0.1487	0.1886	0.1893	0.1889	0.2022
	X	2	0.0591	0.0868	0.0982	0.1345	0.1911	0.2187	0.2306	0.2444	0.2470	0.2451	0.2433	0.2737
	X	4	0.1285	0.1718	0.1937	0.2345	0.2287	0.2537	0.2626	0.2717	0.2833	0.2763	0.2728	0.3017
SDH		0	0.1296	0.1732	0.2245	0.3058	0.1746	0.2115	0.2033	0.2153	0.1659	0.1659	0.2519	0.2656
PHashing (SDH)	X	1	0.0092	0.0198	0.0219	0.0301	0.1181	0.1205	0.1210	0.1286	0.1665	0.1662	0.1690	0.1691
	X	2	0.0493	0.0769	0.1012	0.1435	0.1525	0.1775	0.1740	0.1912	0.1662	0.1656	0.1798	0.1797
	X	4	0.1242	0.1688	0.2117	0.2981	0.1710	0.2069	0.1994	0.2126	0.1659	0.1657	0.2266	0.2240
DPSH		0	0.2591	0.5038	0.5980	0.6970	0.6872	0.7024	0.7281	0.7437	0.4169	0.4911	0.5103	0.5438
PHashing (DPSH)	X	1	0.0204	0.0326	0.0460	0.0751	0.1920	0.2418	0.2816	0.3616	0.2358	0.2724	0.2999	0.3269
	X	2	0.1196	0.3108	0.4307	0.5819	0.5720	0.6350	0.6700	0.7059	0.3704	0.4503	0.4994	0.5011
	X	4	0.2358	0.4797	0.5782	0.6851	0.6752	0.6991	0.7196	0.7386	0.4119	0.4968	0.5085	0.5123
DSH		0	0.6773	0.7164	0.7053	0.7202	0.7230	0.7644	0.7746	0.7920	0.5127	0.5093	0.5184	0.5168
PHashing (DSH)	X	1	0.0245	0.0515	0.0671	0.1264	0.1749	0.2903	0.3435	0.4302	0.2687	0.3225	0.3492	0.3864
	X	2	0.3700	0.6262	0.6379	0.6950	0.5264	0.7109	0.7242	0.7411	0.4734	0.4979	0.5039	0.5050
	X	4	0.6555	0.7122	0.6997	0.7173	0.6958	0.7548	0.7579	0.7653	0.5100	0.5078	0.5158	0.5142
HashNet		0	0.3005	0.4903	0.5503	0.6313	0.7261	0.7614	0.7858	0.7950	0.5516	0.5663	0.5723	0.5782
PHashing (HashNet)	X	1	0.0231	0.0320	0.0366	0.0638	0.2206	0.3139	0.3917	0.4925	0.2844	0.3325	0.3570	0.3963
	X	2	0.1536	0.3113	0.3752	0.5054	0.6413	0.7139	0.7489	0.7634	0.5083	0.5337	0.5399	0.5512
	X	4	0.2743	0.4684	0.5308	0.6181	0.7162	0.7491	0.7710	0.7784	0.5469	0.5623	0.5681	0.5745
ADSH		0	0.0630	0.1696	0.3029	0.5720	0.6599	0.7413	0.7590	0.7672	0.5691	0.5938	0.6022	0.6137
PHashing (ADSH)	X	1	0.0111	0.0129	0.0249	0.0618	0.1745	0.2802	0.3503	0.3980	0.2738	0.3196	0.3460	0.3750
	X	2	0.3594	0.0745	0.0648	0.2149	0.5119	0.6969	0.7260	0.7432	0.5031	0.5372	0.5475	0.5249
	X	4	0.5834	0.1090	0.2139	0.4864	0.6379	0.7360	0.7519	0.7645	0.5596	0.5797	0.5877	0.5581

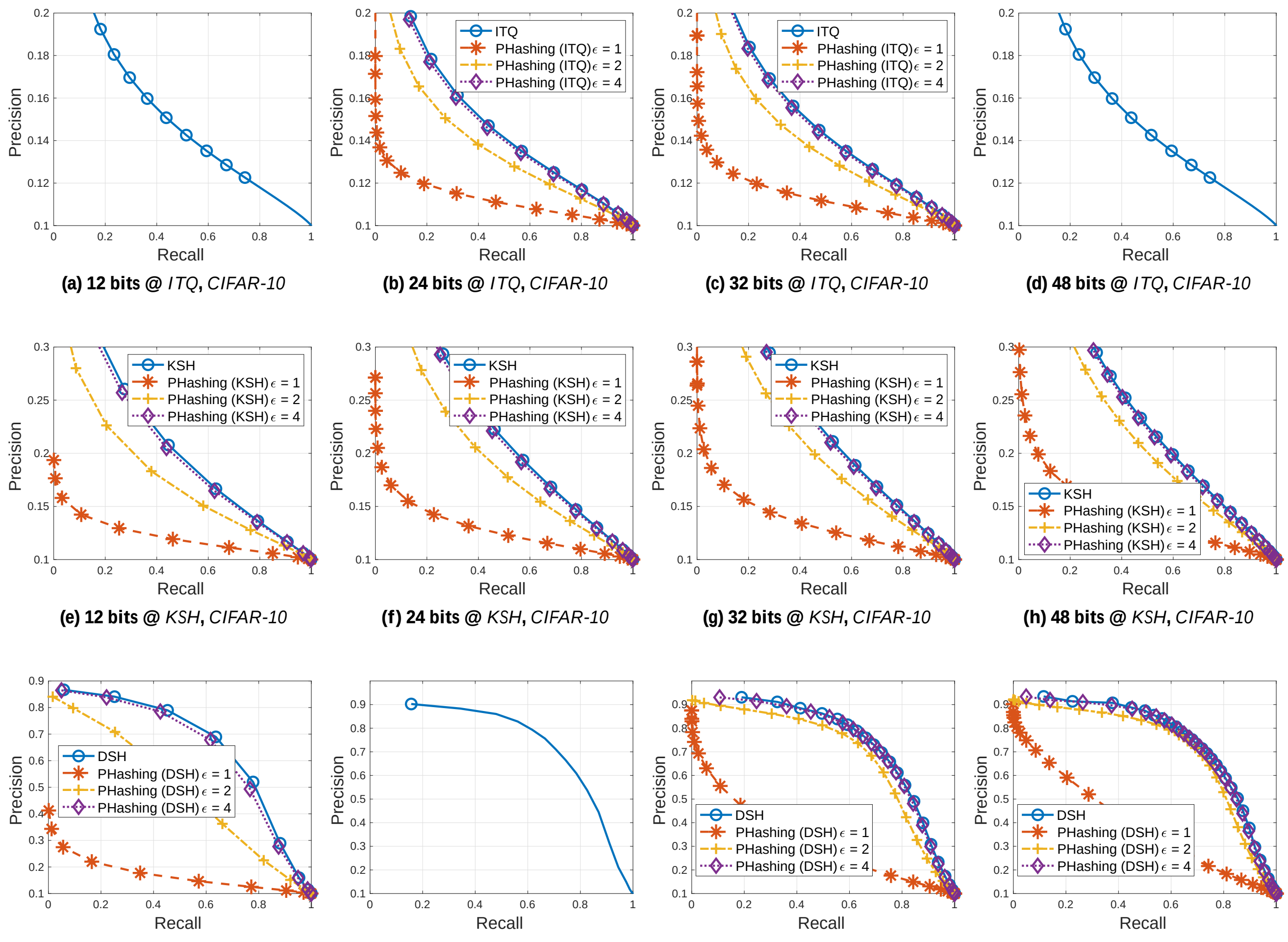
Table 2: Comparisons of mAP w.r.t. the different lengths of bits on *ImageNet*, *CIFAR-10*, *NUS-WIDE* of our proposed PITQ and non-private ITQ method.

Method	Private?	ϵ	ImageNet				CIFAR-10				NUS-WIDE			
			12 bits	24 bits	32 bits	48 bits	12 bits	24 bits	32 bits	48 bits	112 bits	24 bits	32 bits	48 bits
ITQ		0	0.2917	0.4929	0.5586	0.6337	0.1544	0.1607	0.1630	0.1656	0.3189	0.3264	0.3277	0.3322
PITQ	X	0.25	0.2921	0.4887	0.5628	0.6398	0.1543	0.1608	0.1625	0.1655	0.3221	0.3257	0.3275	0.3311
	X	0.5	0.2918	0.4880	0.5633	0.6406	0.1544	0.1608	0.1626	0.1657	0.3223	0.3257	0.3277	0.3313
	X	1	0.2922	0.4883	0.5598	0.6389	0.1543	0.1607	0.1626	0.1659	0.3217	0.3256	0.3276	0.3310
	X	4	0.2928	0.4881	0.5663	0.6374	0.1538	0.1610	0.1624	0.1659	0.3224	0.3257	0.3273	0.3306

4.1 Datasets and Evaluation Details

We evaluate the performance of our proposed methods on five vision datasets, three generic datasets and two fine-grained datasets.

- *ImageNet [11] (Generic Dataset):* ImageNet is a classical benchmark image dataset and contains over 1.2M images in the training set and 50K images in the validation set. Each image has a single label of one of the 1,000 categories. We



(a) 12 bits @ *ITQ*, *CIFAR-10* **(b) 24 bits @ *ITQ*, *CIFAR-10*** **(c) 32 bits @ *ITQ*, *CIFAR-10*** **(d) 48 bits @ *ITQ*, *CIFAR-10***

(e) 12 bits @ *KSH*, *CIFAR-10* **(f) 24 bits @ *KSH*, *CIFAR-10*** **(g) 32 bits @ *KSH*, *CIFAR-10*** **(h) 48 bits @ *KSH*, *CIFAR-10***

(i) 12 bits @ *DSH*, *CIFAR-10* **(j) 24 bits @ *DSH*, *CIFAR-10*** **(k) 32 bits @ *DSH*, *CIFAR-10*** **(l) 48 bits @ *DSH*, *CIFAR-10***

Figure 2: Performance of precision-recall curves on *CIFAR-10* comparing several non-private and private methods, *i.e.*, ITQ and PHashing (ITQ), KSH and PHashing (KSH), DSH and PHashing (DSH). The four sub-figures in each row are the precision-recall curves for 12 bits, 24 bits, 32 bits, and 48 bits, respectively. Best view in color.

randomly select 100 categories, use all the images of these categories in the training set as the database, and use all the images in the validation set as the query.

- *CIFAR-10 [26] (Generic Dataset):* This dataset contains 60,000 32 × 32 color images. Each image belongs to one of the ten classes. We follow the protocol proposed in [26], where 10,000 images (1000 images per class) in the test set is used for evaluation, with the remaining 50,000 images as database points.
- *NUS-WIDE [10] (Generic Dataset):* This is a public Web image dataset which contains 269,648 images downloaded from Flickr.com. Each image is manually annotated by multi-classes of 81 categories for evaluating retrieval models. We randomly sample 5,000 images as queries, with the remaining images as the database.
- *CUB [48] (Fine-grained Dataset):* This is a fine-grained dataset focusing on birds with 11,788 images for 200 various birds species. Database and test sets have 5,994 and 5,794 images.
- *Aircraft [36] (Fine-grained Dataset):* Aircraft contains 10,000 images of aircraft, with 100 images for each of 100 different aircraft model variants. We also follow the standard split in [36].

For the above datasets, two images are treated as a ground-truth similar pair if they share at least one label. We evaluate the search performance by adopting two evaluation metrics: mean Average Precision (mAP) and Precision-recall Curves (PR Curves) based on lookup. PR Curves are obtained by varying the Hamming radius from 0 to c with the step-size of 1 to demonstrate the hash lookup procedure. All the data are reported with average values running five times.

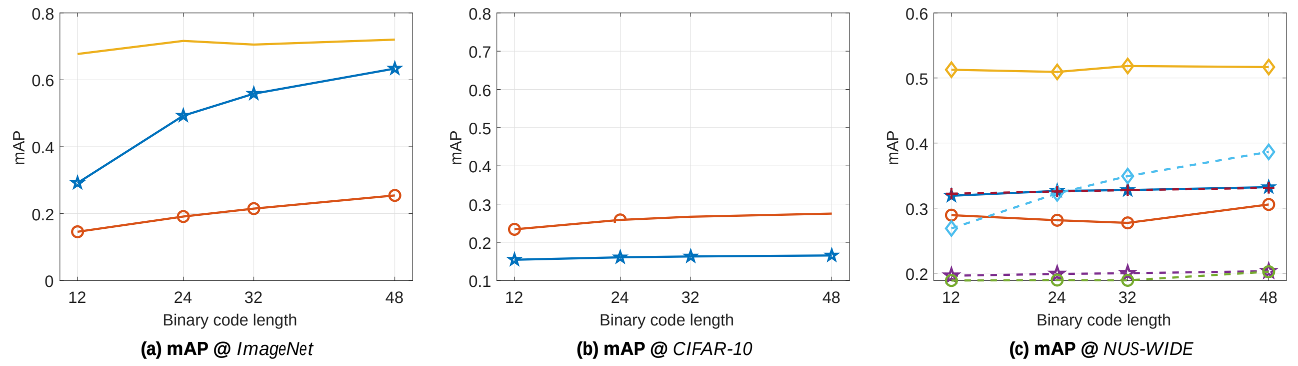


Figure 3: mAP comparison on *ImageNet*, *CIFAR-10*, and *NUS-WIDE* with $\epsilon = 1$ for some private methods. For a clear presentation, we omit the legend of Fig. 3(a) and Fig. 3(c) whose legends are the same as Fig. 3(b) . Best view in color.

We use several non-private state-of-the-art hashing methods to compare the performance of PHashing, including shallow methods, *i.e.*, LSH [17], ITQ [18], SH [51], SDH [42] and KSH [33], and deep supervised methods, *i.e.*, DSH [32], DPSH [30], HashNet [5], and ADSH [24]. To simplify the notation, we denote the hashing algorithm extended by PHashing as PHashing ¹·º. For example, PHashing (ITQ) represents the private version of ITQ extended by PHashing, while H ¹·, ·º is ITQ. For deep hashing methods, we use raw images resized to 224×224 as inputs. For traditional shallow methods, we extract 4096-dimensional deep features by the VGG-D [45] model pre-trained with ImageNet to conduct fair comparisons. Besides, for all the non-private hashing methods, we employ hyper-parameters introduced in their papers.

4.2 Performance Drop Caused by Privacy Protection – Parameter Sensitivity

To validate the cost of privacy protection, we choose ϵ from the set f1, 2, 4g of PHashing and f0.25, 0.5, 1, 4g of PITQ. The mAP of dierent methods on three generic datasets is presented in Tab. 1 and Tab. 2 with code-lengths of 12 bits, 24 bits, 32 bits, and 48 bits, respectively. PR Curves of some methods are shown in Fig. 2. The left mAP results and PR Curves are presented in Appendix due to the limitation of space.

Search accuracy: In Tab. 1, the results demonstrate that the search accuracy is inversely proportional to the privacy parameter ϵ, which is consistent with the theoretical analysis. And as the parameter ϵ increases, accuracies of PHashing are approaching non-private accuracies. The search accuracy in Tab. 2 shows that distributing the noise to several iterations signicantly improves the search performance, as our PITQ achieves equal performances with the non-private ITQ and outperforms PHashing (ITQ).

PR Curves: Fig. 2 presents several precision-recall curves for PHashing using ITQ, KSH, and DSH on *CIFAR-10*. As the curves show, the private curves are approaching the non-private curves with ϵ increasing. The private curves of $\epsilon = 4$ are almost the same as non-private curves.

4.3 Search Performance

To validate the search performance, we set $\epsilon = 1$ for PHashing and PITQ. The mAP of dierent methods on three generic datasets is presented in Fig. 3 with the code-lengths of 12 bits, 24 bits, 32 bits, and 48 bits, respectively. Due to the limitation of space, the mAP comparison of ne-grained datasets and PR Curves of all ve datasets are presented in Appendix.

Search accuracy: In Fig. 3, PITQ outperforms PHashing (ITQ). As PITQ distributes the noise to each iteration, the performance drop is reduced and PITQ achieves satisfying performance. Due to the power of supervised learning and deep learning, PITQ is worse than some deep supervised private hashing methods, which is reasonable.

5 CONCLUSION

In this paper, we presented a novel noise mechanism and two private algorithms for the *private hashing* task. One of the key contributions was the novel *Random Flipping* algorithm designed for the binary situation, in other words, hashing scenario, which ensured ϵ-dierential privacy. Besides, our rst algorithm, PHashing, showed its ability to extend any non-private hashing methods to preserve ϵ-dierential privacy with acceptable performance drops. Also, our second algorithm, PITQ, improved the search accuracy by distributing the noise generated by Random Flipping to each iteration while achieving equal performances comparing with the non-private ITQ, as the noise could practically help to nd a better local optimum with ϵ-dierential privacy. Experimental results on diverse vision datasets showed the eectiveness of our methods. In the future, we would like to explore novel private hashing methods under the supervised setting and reduce performance drop.

ACKNOWLEDGMENTS

This work was partially supported by NSFC-NRF Joint Research Project (61861146001), the Fundamental Research Funds for the Central Universities (14380074), and the Collaborative Innovation Center of Novel Software Technology and Industrialization.

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep Learning with Di erential Privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. Association for Computing Machinery, New York, NY, USA, 308–318. https://doi.org/10.1145/2976749.2978318

[2] Naman Agarwal and Karan Singh. 2017. The Price of Di erential Privacy for Online Learning. In *Proceedings of the 34th International Conference on Machine Learning*. PMLR, International Convention Centre, Sydney, Australia, 32–40.

[3] Alexandr Andoni and Piotr Indyk. 2008. Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions. *Commun. ACM* 51, 1 (Jan. 2008), 117–122. https://doi.org/10.1145/1327452.1327494

[4] Alexandr Andoni and Ilya Razenshteyn. 2015. Optimal Data-Dependent Hashing for Approximate Near Neighbors. In *Proceedings of the 47th Annual ACM Symposium on Theory of Computing*. Association for Computing Machinery, New York, NY, USA, 793–801. https://doi.org/10.1145/2746539.2746553

[5] Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Philip S. Yu. 2017. HashNet: Deep Learning to Hash by Continuation. In *The IEEE International Conference on Computer Vision*. 5608–5617.

[6] Zhangjie Cao, Ziping Sun, Mingsheng Long, Jianmin Wang, and Philip S. Yu. 2018.

[44] Reza Shokri and Vitaly Shmatikov. 2015. Privacy-Preserving Deep Learning. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. Association for Computing Machinery, New York, NY, USA, 1310–1321. https://doi.org/10.1145/2810103.2813687

[45] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556* (2014).

[46] Ge Song and Xiaoyang Tan. 2017. Hierarchical deep hashing for image retrieval. *Frontiers of Computer Science* 11, 2 (2017), 253–265. https://doi.org/10.1007/s11704-017-6537-3

[47] Jingkuan Song, Lianli Gao, Yan Yan, Dongxiang Zhang, and Nicu Sebe. 2015. Supervised Hashing with Pseudo Labels for Scalable Multimedia Retrieval. In *Proceedings of the 23rd ACM International Conference on Multimedia*. Association for Computing Machinery, New York, NY, USA, 827–830. https://doi.org/10.1145/2733373.2806341

[48] Catherine Wah, Steve Branson, Peter Welinder, and Serge Belongie Pietro Perona. 2011. *The Caltech-UCSD Birds-200-2011 Dataset*. Technical Report CNS-TR-2011-001. California Institute of Technology.

[49] Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. 2010. Semi-supervised hashing for scalable image retrieval. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 3424–3431.

[50] Yimu Wang, Renjie Song, Xiu-Shen Wei, and Lijun Zhang. 2020. An Adversarial Domain Adaptation Network For Cross-Domain Fine-Grained Recognition. In *2020 IEEE Winter Conference on Applications of Computer Vision*. 1217–1225.

[51] Yair Weiss, Antonio Torralba, and Rob Fergus. 2009. Spectral Hashing. In *Advances in Neural Information Processing Systems 21*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou (Eds.). Curran Associates, Inc., 1753–1760.

[52] Liang Xie, Jialie Shen, Jungong Han, Lei Zhu, and Ling Shao. 2017. Dynamic Multi-View Hashing for Online Image Retrieval. In *Proceedings of the 26th International Joint Conference on Arti cial Intelligence*. AAAI Press, 3133–3139.

[53] Xinyu Yan, Lijun Zhang, and Wu-Jun Li. 2017. Semi-Supervised Deep Hashing with a Bipartite Graph. In *Proceedings of the 26th International Joint Conference on Arti cial Intelligence*. 3238–3244.

[54] Chengyuan Zhang, Lei Zhu, Shichao Zhang, and Weiren Yu. 2020. TDHPPIR: An E cient Deep Hashing Based Privacy-Preserving Image Retrieval Method. *Neurocomputing* 406 (2020), 386 – 398. https://doi.org/10.1016/j.neucom.2019.11.119

[55] Jun Zhang, Zhenjie Zhang, Xiaokui Xiao, Yin Yang, and Marianne Winslett. 2012. Functional Mechanism: Regression Analysis under Di erential Privacy. *Proceedings of the VLDB Endowment* 5, 11 (July 2012), 1364–1375. https://doi.org/10.14778/2350229.2350253

[56] Ruimao Zhang, Liang Lin, Rui Zhang, Wangmeng Zuo, and Lei Zhang. 2015. Bit-Scalable Deep Hashing With Regularized Similarity Learning for Image Retrieval and Person Re-Identi cation. *IEEE Transactions on Image Processing* 24, 12 (2015), 4766–4779.