# BiasHash: A Bayesian Hashing Framework for Image Retrieval

Maria Pegia*, Anastasia Moumtzidou*, Ilias Gialampoukidis*, Björn Þór Jónsson†,
Stefanos Vrochidis* and Ioannis Kompatsiaris*
*Information Technologies Institute Centre of Research & Technology - Hellas, Greece
Email: {mpegia, moumtzid, heliasgj, stefanos, ikom}@iti.gr
†IT University of Copenhagen, Denmark
Email: bjth@itu.dk

***Abstract*—Hashing methods have been widely used for similarity search in multimedia due to its low memory requirements and efficient and scalable search. Recently, an approach called Semantic Preserving Hashing (SePH) has been proposed, which uses the semantic probabilities of training data, approximates them with the learnt hash codes and then uses the kernel logistic regression for learning the projection of features to the learnt hash codes. In this paper, we extend this method using a Bayesian framework to learn these projection functions, motivated by the probability distribution that visual features tend to approximate. The proposed Bayesian ridge-based Semantic Preserving Hashing (BiasHash) approach is shown to outperform seven state-of-the-art methods on three benchmark datasets.**

## I. Introduction

Image retrieval is the field of study concerned with searching, browsing and retrieving digital images from database. The main challenges are (a) the semantic gap between the low-level feature representing and high-level semantics in the images, and (b) the curse of dimensionality, since visual descriptors usually have hundreds or even thousands of dimensions. A hash-based index offers reduced storage, by storing only compact binary codes in the index, and constant average response time. Therefore, we focus on hashing methods in order to use fast search through hash-indexed data instead of inefficient exhaustive search, and in parallel we aim to minimize the aforementioned semantic gap for effective image retrieval.

Hashing approaches are categorized into single-view [1], [2], [3], [4], [5], [6], [7] and multi-view [8], [9], [10], [11], [12], [13], [14], [15]. The former approaches use only one view, while the latter approaches importantly support many views/modalities (text, image, video). Another categorisation is based on the nature of the hashing functions used to generate the binary codes. Early approaches to hash-based indexing used manually-tuned hashing functions (e.g., [16]), but more recent hashing approaches use either unsupervised learning [3], [7] or supervised learning [9], [4], [10], [11], [12], [13], [17], [5], [6], [14], [15] methods to generate the hash function.

A relatively recent supervised hashing method, Semantic Preserving Hashing (SePH), has been proposed in [9]. It generates one unified hash code for all observed views of any instance. It transforms the semantic affinities of training data into a probability distribution and aims to approximate it with another one in Hamming space, via minimizing their Kullback-Leibler divergence [18]. Then it uses kernel logistic regression to learn the hash codes.

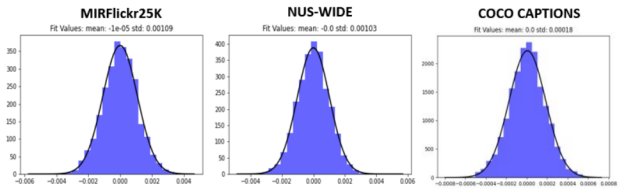


Fig. 1: Histogram of VGG16 vectors for MIRFlickr25K, NUS-WIDE and COCO CAPTIONS datasets.

State-of-the-art hashing approaches require significant training data to learn hash functions, before they can be applied. We observe that modern image features for typical benchmark collections follow statistical distributions closely; Figure 1 show that features from pre-trained VGG-16 network closely follow the Gaussian distribution [19], for the MIR-Flickr25K, NUS-WIDE, and COCO CAPTIONS collections, respectively. However, if the training set is small, or the training data does not adequately represent the image collection, then current hashing approaches will perform sub-optimally.

A major advantage of Bayesian estimation is that one can incorporate the use of a prior, or assumed knowledge about the current state of "beliefs", and how the evidence might update those beliefs. One can use a Bayesian approach in a hashing framework, so that it depends less on the unbalanced dataset and produces compact binary codes. This benefit is much appropriate in supervised hashing methods, where often the datasets are very unbalanced and thus the way of splitting the dataset into query, training and retrieval sets affects the performance of retrieval.

In this work, we therefore propose a Bayesian-based supervised hashing method. Our method preserves the advantages of hashing methods (low memory requirements, low complexity time), while it exploits the advantages of Bayesian framework to learn hash functions, in such a way that the choice of the training data can be considered almost or totally negligible. We expect that it can outperform the existing supervised methods, due to its weak dependency on the splitting of the dataset.

## II. Methodology

Figure 2 shows an overview of the proposed BiasHash framework. In an offline phase, it computes the affinity matrix from training label vectors (Step 1) and the semantic

978-1-6654-7822-9/22/$31.00 ©2022 IEEE

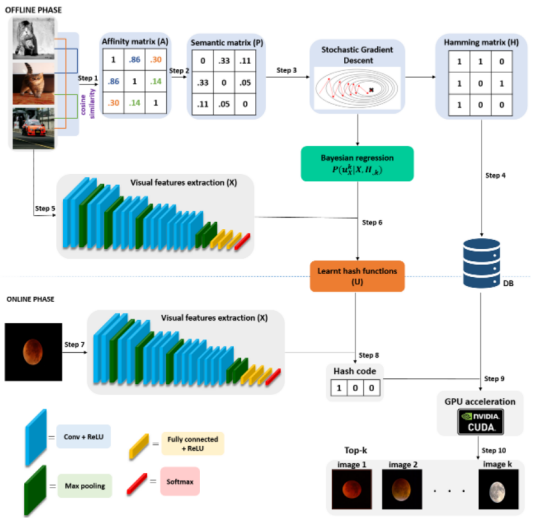


Fig. 2: Overview of the proposed indexing method.

probabilities from affinity values (Step 2). Then it projects these values to the learnt Hamming probabilities solving a minimization problem (Step 3). The Hamming vectors are stored in a database (Step 4). The approach extracts the visual features from training images (Step 5) and learns the hash functions (Step 6) from the visual vectors to Hamming codes using Bayesian ridge regression. In the online phase, for a given query, the approach extracts the visual feature (Step 7) and computes its Hamming code using the learnt hash functions (Step 8). Finally it computes the Hamming distances between query and database codes in GPU (Step 9), ranks the results and returns the top-$k$ most relevant (Step 10).

### A. Notation

Let $\mathcal{O}$ be the training set of size $|\mathcal{O}| = n$, with $O_i$ its $i$-th instance. We define $X \in \mathbb{R}^{n \times d_x}$, $L \in \{0,1\}^{n \times l}$ as the visual features and semantic labels of the training set, respectively, where $l$ is the total number of labels. For each instance $O_i$ we build the visual feature vector $X_{i,.} \in \mathbb{R}^{d_x}$ and the semantic feature vector $L_{i,.} \in \{0,1\}^{l}$. We define $A \in [0,1]^{n \times n}$ as the affinity matrix of the training set. We denote with $H \in \{0,1\}^{n \times d_c}$ the learnt hash codes of the training set. We define each instance of the training set as $H_{i,.} \in \{0,1\}^{d_c}$ of code length $d_c$. Each row of $H$ (i.e., $H_{i,.}$) corresponds to the projection of each semantic instance. The notation $u_k \in \mathbb{R}^{d_x}$ corresponds to the learnt hash function of $k$-th bit, for $1 \leq k \leq d_c$. We denote by $h(.,.)$ the Hamming distance between two hash codes. We denote with $c^X$ the hash codes of respective visual features $X$ and with $c^x_k$ the $k$-th bit of hash code $c^x \in \{0,1\}^{d_c}$ of visual feature $x \in \mathbb{R}^{d_x}$, for $1 \leq k \leq d_c$.

### B. Semantics Preserving Hashing

As the proposed method builds on SePH, we describe that method first, and then outline the differences in the following subsection. As described above, the SePH first computes the affinity matrix $A$ using the cosine similarity of corresponding semantic label vectors $A_{i,j} = \frac{< L_{i,.}, L_{j,.} >}{||L_{i,.}||||L_{j,.}||}$ (1). Then the probabilities are: $p_{i,j} = \frac{A_{i,j}}{\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n} A_{i,j}}$ (2) in semantic space $\mathcal{P}$. From the work of van der Maaten and Hinton [20], we know that a Student t-distribution with one degree of freedom is utilized for transforming each pairwise Hamming distance into a probability. Hence, the $q_{i,j} = \frac{(1+h(H_{i,.},H_{j,.}))^{-1}}{\sum_{k=1}^{n}\sum_{m=1,m\neq k}^{n}(1+h(H_{k,.},H_{m,.}))^{-1}}$ (3) computes the corresponding probabilities $q_{i,j}$ of instances in Hamming space $\mathcal{Q}$.

This framework uses Kullback-Leibler divergence to measure the differences between $\mathcal{Q}$ and $\mathcal{P}$. Therefore, it can learn the optimal binary hash code matrix $H$ of training set by minimizing it, However, this minimization problem is NP-hard [21] and therefore we relax $H$ to a real valued matrix $\hat{H}$ in Eq. (4):

$$\Psi = \min_{\hat{H} \in \mathbb{R}^{n \times d_c}} \sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n} p_{i,j} log\frac{p_{i,j}}{q_{i,j}} + \frac{a}{C}|||\hat{H}| - I||_2^2 \quad (4)$$

with $q_{i,j} = \frac{(1+||\hat{H}_{i,.}-\hat{H}_{j,.}||_2^2)^{-1}}{\sum_{k=1}^{n}\sum_{m=1,m\neq k}^{n}(1+||\hat{H}_{i,.}-\hat{H}_{j,.}||_2^2)^{-1}}$ (5), $\Psi$ is the minimization problem, $I \in \{1\}^{n \times d_c}$, $a$ is a model parameter for weighting quantization loss and $C = n \times d_c$ is a normalization factor to make the parameter tuning for $a$ less affected by the hash code length and the training size.

The minimization problem of Eq. (4) is an unconstrained non-convex optimization problem and thus we can derive only a locally optimal $\hat{H}$. We use stochastic gradient descent [22] to find $\hat{H}$ and it computes a Hamming space matrix $H = sign(\hat{H})$. Then it uses kernel logistic regression [9] to learn the hash function that projects the visual features to derived hash codes $H$. With learnt hash functions for indexing process, the view-specific hash codes of any unseen instance $z_u$ can be predicted.

For each query, the approach measures its Hamming distance from each element in the retrieval set, sorts the retrieval set in ascending order based on this measure, and then picks the top $k$ elements from the ordered retrieval set.

More precisely, we define $c^{\mathcal{X}}$ as the predictive visual hash code and let $c^{\mathcal{X}}_k$, be its $k$-th bit. Then the $c^{\mathcal{X}}_k = sign((\phi(X_{i,.})\hat{\Phi}^t)u^{(k)})$ (6) computes the predicted hash codes using kernel logistic regression (6) for $1 \leq k \leq d_c$ and $u^{(k)} \in \mathbb{R}^{d_x}$. $\Phi$ is the kernel feature matrix and $\phi(X_{i,.})$ is the transformation of visual feature $X_{i,.}$ in the Reproducing Kernel Hilbert Space (RKHS).

Similar to most hashing methods, SePH uses Hamming distance to perform retrieval for query hash code $H_q$ from the retrieval hash codes:

$$h(H_q, H_i) = bit\_count(H_q \oplus H_i) \quad (7)$$

where $\oplus$ denotes the XOR operation between the bits of $H_q$ and $H_i$, and $bit\_count$ counts the number of 1 in the binary XOR result. After that we rank all instances in the retrieval set based on their Hamming distances in an ascending order and take the top ones.

### C. Hashing with Bayesian Ridge Regression

In our proposed framework, we take as input the image features, using some descriptor, and compute the affinity matrix,

**Algorithm II.1** BiasHash method

**Input:** $X \in \mathbb{R}^{n \times d_x}$, $L = \in \{0,1\}^{n \times l}$, $\lambda_1 = \lambda_2 = \alpha_1 = \alpha_2 = e^{-6}$, $d_c$, tol = $e^{-3}$, maxIter = 300
**Output** posterior mean $u_k^X \in \mathbb{R}^{d_x}$

1. Compute affinity matrix $A$ using Eq. (1).
2. Compute semantic space matrix $\mathcal{P}$ using Eq. (2).
3. Compute relaxed Hamming space matrix $\hat{H}$ using Eq. (4).
4. Set Hamming space matrix $H = sign(\hat{H})$.
5. for $k$ in range($d_c$):
6. Set $y = H_{.,k}$.
7. Compute $\alpha = \dfrac{1}{var(y) + 2.22e^{-16}}$, $\lambda = 1, z = X^t y$.
8. Perform SVD on $X$ and get $U$, $S$, $V$.
9. Compute eigenvalues eigVal = $S^2$.
10. Initialize posterior mean $u^k_{X_{(old)}}$ = None.
11. for i in range(maxIter):
12. Update posterior mean $u_X^k = (\dfrac{\lambda}{\alpha} I_{d_x} + X^t X)^{-1} X^t y$, from Eq. (9) and $(\dfrac{\lambda}{\alpha} I_{d_x} + X^t X)^{-1} X^t$, $y$ are the likelihood and the prior mean, respectively.
13. Compute root mean square deviation rmse = $||y - X u_X^k||_2^2$.
14. Compute $M = \dfrac{\alpha \times eigVal}{\lambda + \alpha \times eigVal}$.
15. Update $\gamma = \sum_{i,j} M_{i,j}$, $\lambda = \dfrac{\gamma + 2\lambda_1}{||u_X^k||_2^2 + 2\lambda_2}$, $\alpha = \dfrac{n - \gamma + 2\alpha_1}{rmse + 2\alpha_2}$
16. If $i \neq 0$ and $||u^k_{X_{(old)}} - u_X^k|| < tol$:
17. break
18. Compute $u_X^k = (\dfrac{\lambda}{\alpha} I_{d_x} + X^t X)^{-1} X^t y$
19. Compute bit-by-bit hash codes using Eq. (10).

the semantic space matrix, and the Hamming space matrix, as described above. Inspired by motivation in the introduction, however, we replace the predictive model of SePH (Equation (6)) with Bayesian Ridge Regression [23]. It is a linear model that uses probability distributions rather than point estimation of linear regression. The linear regression minimizes loss, while the Bayesian version maximizes the posterior probability by fitting a probabilistic model. In particular, we have a linear model:

$$\begin{aligned} H_{.,k} &= X u_X^k + \epsilon \\ \epsilon &\sim N(0_n, \sigma^2 I_n) \\ H_{.,k} &\in \{0,1\}^n \end{aligned} \tag{8}$$

where $H_{.,k}$ is the $k$-th column of learnt hash codes $H$, and $X$ are the visual features. We can also formulate it, considering probability distributions and prior before seeing all data (only train set), as: $H_{.,k} \sim N(X u_X^k, \sigma^2 I_n)$.

We recall the Bayes' formula

$$P(u_X^k | X, H_{.,k}) = \frac{P(X | u_X^k, H_{.,k}) * P(u_X^k | H_{.,k})}{P(X, H_{.,k})} \tag{9}$$

that combines posterior and prior probability: where $P(u_X^k | X, H_{.,k})$, $P(u_X^k | H_{.,k})$ are the posterior and prior probability, respectively, whereas $P(X | u_X^k, H_{.,k})$, $P(X, H_{.,k})$ are the corresponding likelihood and normalization. Then we compute posterior mean for each modality using the iterative method of Tipping algorithm [23], [24] based on parameters updates proposed by MacKay and set it to $u_X^k$.

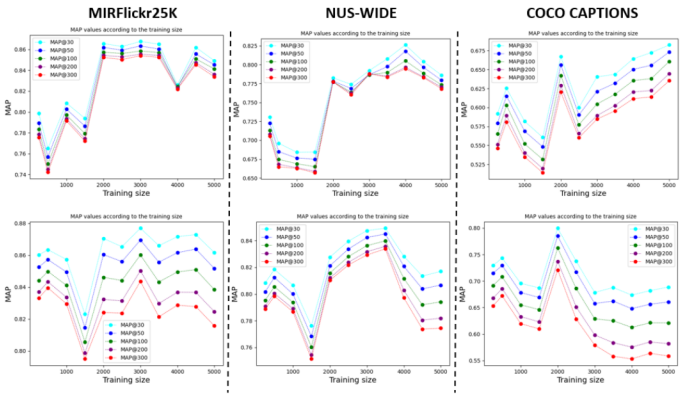


Fig. 3: MAP according to training size of MIRFlickr25K, NUS-WIDE and COCO CAPTIONS datasets for 16 and 128 bits.

Algorithm II.1 outlines our method. Lines 6-18 describe the iterative method of Tipping settings and the way of parameters' update by MacKay. We compute hash codes $c^X \in \{0,1\}^{n \times d_c}$ using features and learnt hash functions. Each bit $k$ = 1, ..., $d_c$ of hash code $c_k^{\mathcal{X}} = sign(X_{i,.} u^{(k)})$ (10) for a given visual feature $x$ After Step 19, for a given query image, we extract its hash codes $H_q$ and based on Eq. 7, we compute similarity scores with the indexed images in the database.

## III. Experiments

### A. Datasets

We use for our experiments MIRFlickr25K [25], NUS-WIDE and COCO CAPTIONS. From the retrieval set we pick some instances to form the training set for MIR-Flickr25K (27.2%), NUS-WIDE (4.28%) and COCO CAP-TIONS (7.31%). For each image we use a 4096-D vector from the fc-7 layer of VGG-16 pre-trained network on ImageNet. We choose for training 2,000, 3500, 2000 images (Figure 3) from MIRFlickr25K, NUS-WIDE and COCO CAPTIONS datasets, respectively.

### B. Settings

In our experiments, the affinity matrix $A$ is derived from the cosine similarity between semantic labels of training data. Model parameter $a = 0.01$ for Eq. 4 as [9]. We compute hash codes of bit length $d_c$ = 16, 32, 64, 128. As VGG-16 is used for visual features, we set $d_x = 4096$ for all datasets. We use four hyperparameters for Bayesian regression, $a_1, a_2, \lambda_1, \lambda_2$ of the gamma prior distributions over $a$ and $\lambda$ and set them to $10^{-6}$ based on [23], [24]. We have designed a method for solving a uni-modal problem (image retrieval), but most of the baselines are cross-modal methods (except HCOH). Therefore we set their corresponding parameters to zero to avoid fusion.

We choose randomly 2,000, images from MIRFlickr25K and COCO and 2100 images for NUS-WIDE and COCO CAP-TIONS datasets, respectively. We use mean Average Retrieval (mAP) (11) to measure the retrieval performance of methods.

TABLE I: Experiments of each method for MIRFLickr25K, NUS-WIDE and COCO CAPTIONS for code length 16, 32, 64 and 128 bit.

| Dataset | Method | 16bit | 32bit | 64bit | 128bit |
|---|---|---|---|---|---|
| MIRFlickr25k | SSAH [10] | 0.82700* | 0.83000* | 0.84000* | 0.81000* |
| | KDLFH [11] | 0.82557 | 0.83014 | 0.88686* | 0.90118 |
| | GPSH [12] | 0.81435* | 0.83133 | 0.84458* | 0.85211 |
| | MTFH [13] | 0.82088 | 0.83124 | 0.86926* | 0.88076 |
| | HCOH [4] | 0.57916* | 0.57978* | 0.58082* | 0.58196* |
| | SePH [9] | 0.66474* | 0.67911* | 0.68001* | 0.68645* |
| | CSQ [17] | 0.82000* | 0.83000* | 0.87800* | 0.88500 |
| | BiasHash | **0.82758** | **0.83136** | **0.89281** | **0.90765** |
| NUS-WIDE | SSAH [10] | 0.71000* | 0.72300* | 0.71000* | 0.69500* |
| | KDLFH [11] | 0.80827 | 0.84113 | 0.80119* | 0.84119* |
| | GPSH [12] | 0.81633 | 0.84165 | 0.80470* | 0.83027* |
| | MTFH [13] | 0.80453 | 0.83989 | 0.80974* | 0.84364* |
| | HCOH [4] | 0.50531* | 0.50596* | 0.50482* | 0.50599* |
| | SePH [9] | 0.68084* | 0.68970* | 0.60200* | 0.53002* |
| | CSQ [17] | 0.80200* | 0.81600* | 0.81300* | 0.81800* |
| | BiasHash | **0.82264** | **0.84234** | **0.81455** | **0.86139** |
| COCO CAPTIONS | SSAH [10] | 0.35900* | 0.35700* | 0.35800* | 0.35700* |
| | KDLFH [11] | 0.63071* | 0.65689* | 0.72430* | 0.75280* |
| | GPSH [12] | 0.64686* | 0.71350* | 0.70037* | 0.80590* |
| | MTFH [13] | 0.20462* | 0.23702* | 0.20482* | 0.21321* |
| | HCOH [4] | 0.33401* | 0.33645* | 0.34014* | 0.35439* |
| | SePH [9] | 0.63601* | 0.69357* | 0.60312* | 0.58722* |
| | CSQ [17] | 0.65410* | 0.71380* | 0.7211* | 0.81300 |
| | BiasHash | **0.65760** | **0.76800** | **0.76659** | **0.82223** |

$$mAP = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{m_i}\sum_{j=1}^{m_i} precision(R_{j,i}) \qquad (11)$$

where $Q$ is the query set. The term of inner sum denotes the average precision of the $i$-th query, where $m_i$ is the number of its ground-truth relevant instances [26], [8] in the retrieval set, $R_{i,j}$ is a subset of its ranked retrieval result consisting of instances from the top one to the $j$-th ground-truth relevant one, and precision($R_{i,j}$) measures the precision value in $R_{i,j}$.

The proposed BiasHash method is implemented in Python 3.6.9, which is powered by a workstation with Intel Xeon Silver 4210 CPU (2.20 GHz, 10 cores, and 125GB RAM) with GeForce RTX2080 Ti TURBO 11GB GDDR6 NVIDIA GeForce and 18.04.5 LTS Ubuntu software. For more efficient search for nearest-neighbors we perform Steps 9 and 10 of the retrieval process on a GPU, to avoid linear complexity with the number of elements from the database. For the corresponding implementation we use the cuda-knn method[1], initially introduced by [27]. We compare our approach with seven state of the art methods, as also referred in Section I, with online open-source implementations available SSAH[2], KDLFH[3], GPSH[4], MTFH[5], HCOH[6], SePH and CSQ[7].

### C. Results

Table I shows the mAP for the proposed BiasHash approach, compared to the seven state-of-the-art methods for different hash code lengths (length = 16, 32, 64, 128 bit) in all three benchmark datasets. The symbol $*$ indicates that the mAP value the BiasHash method is statistically significant compared to the corresponding method with symbol $*$, using a t-test [28] to measure the significance between the results. We observe that as the hash code length increases, the performance of the BiasHash mainly increases, reflecting its capability of utilizing longer hash codes to better preserve semantic information, but with a memory cost. Only the CSQ baseline approaches the performance and stability of BiasHash, yet BiasHash outperforms CSQ with statistical significance in 9 of 12 settings.

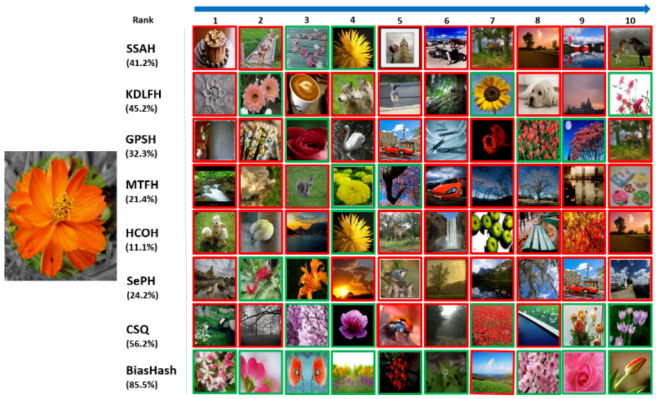


Fig. 4: Retrieval performance of the proposed BiasHash and compared methods on the MIRFlickr25K benchmark dataset with different hash code lengths.

Figure 4 shows the retrieved results for one query from the MIRFlickr25K dataset. The queries were selected as the most representative from each class. The first column has the query, followed by the retrieved images. Each row of the tableau corresponds to the returned results of one method, with the name of the method and the percentages of overall average precision at 100 given at the start of each row. We use a green corner, with a check mark, to indicate that the retrieved image is relevant to the query and a red corner, with an X, to indicate that the retrieved image is not relevant. As the figure shows, the Bayesian method (BiasHash) returns more relevant and highly ranked retrieved images for this given query image.

## IV. Conclusion

In this work we have modified the Semantic Preserving Hashing method by using Bayesian regression as the utilized predictive model. The advantage of Bayesian Regression is the adaption of data at hand from their statistical properties and the ability to include regularization parameters in the estimation process. We show that the proposed BiasHash method outperforms seven state of the art methods over three benchmark datasets. In the future, we plan to compare with more recent methods and to validate our framework in domain-specific and multi-spectral image collections.


## Acknowledgment

This work was supported by the EU's Horizon 2020 research and innovation programme under grant agreements H2020-883302 ISOLA and H2020-101004152 CALLISTO.


[1]https://github.com/vincentfpgarcia/kNN-CUDA
[2]https://github.com/lelan-li/SSAH
[3]https://github.com/jiangqy/DLFH-TIP2019
[4]https://github.com/devraj89/GPSH-algorithm
[5]https://github.com/starxliu/MTFH
[6]https://github.com/TreezzZ/HCOH_Pytorch.git
[7]https://github.com/yuanli2333/Hadamard-Matrix-for-hashing

## REFERENCES


[1] D. Zhang, J. Wang, D. Cai, and J. Lu, "Self-taught hashing for fast similarity search," *In Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval,*, pp. 18 – 25, Jul. 2010. [Online]. Available: https://doi.org/10.1145/1835449.1835455

[2] G. Lin, C. Shen, Q. Shi, A. van den Hengel, and D. Suter, "Fast supervised hashing with decision trees for high-dimensional data," *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1963–1970, Jun. 2014. [Online]. Available: https://doi.org/10.1109/CVPR.2014.253

[3] Y. Zhen, Y. Gao, D. Y. Yeung, H. Zha, and X. Li, "Spectral multimodal hashing and its application to multimedia retrieval," *IEEE Transactions on Cybernetics*, vol. 46, no. 1, pp. 27 – 38, Jan. 2016. [Online]. Available: https://doi.org/10.1109/TCYB.2015.2392052

[4] M. Lin, R. Ji, H. Liu, and Y. Wu, "Supervised online hashing via hadamard codebook learning," *Proceedings of the 26th ACM international conference on Multimedia*, p. 1635–1643, Oct. 2018. [Online]. Available: https://doi.org/10.1145/3240508.3240519

[5] H. Fu, Y. Li, H. Zhang, J. Liu, and T. Yao, "Rank-embedded hashing for large-scale image retrieval," *Proceedings of the 2020 International Conference on Multimedia Retrieval*, pp. 563–570, Jun. 2020. [Online]. Available: https://doi.org/10.1145/3372278.3390716

[6] Y. Xie, Y. Liu, Y. Wang, L. Gao, P. Wang, and K. Zhou, "Label-attended hashing for multi-label image retrieval," *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20)*, pp. 955–962, 2020. [Online]. Available: https://doi.org/10.24963/ijcai.2020/133

[7] H. Cui, L. Zhu, J. Li, Z. Cheng, and Z. Zhang, "Two-pronged strategy: Lightweight augmented graph network hashing for scalable image retrieval," *MM '21: Proceedings of the 29th ACM International Conference on Multimedia*, Aug. 2021. [Online]. Available: https://doi.org/10.1145/3474085.3475605

[8] J. Zhou, G. Ding, and Y. Guo, "Latent semantic sparse hashing for cross-modal similarity search," *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, p. 415–424, Jul. 2014. [Online]. Available: https://doi.org/10.1145/2600428.2609610

[9] Z. Lin, G. Ding, M. Hu, and J. Wang, "Semantics-preserving hashing for cross-view retrieval," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3864–3872, Jun. 2015. [Online]. Available: https://doi.org/10.1109/CVPR.2015.7299011

[10] C. Li, C. Deng, N. Li, W. Liu, X. Gao, and D. Tao, "Self-supervised adversarial hashing networks for cross-modal retrieval," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4242–4251, Jun. 2018. [Online]. Available: https://doi.org/10.1109/CVPR.2018.00446

[11] Q.-Y. Jiang and W.-J. Li., "Discrete latent factor model for cross-modal hashing," *IEEE Transactions on Image Processing*, vol. 28, no. 7, pp. 3490–3501, Jul. 2018. [Online]. Available: https://doi.org/10.1109/TIP.2019.2897944

[12] D. Mandal, K. N. Chaudhury, and S. Biswas, "Generalizes samantic preserving hashing for n-label cross-modal retrieval," *IEEE Transactions on Image Processing*, vol. 28, no. 1, pp. 102 – 112, Jan. 2018. [Online]. Available: https://doi.org/10.1109/TIP.2018.2863040

[13] X. Liu, Z. Hu, H. Ling, and Y. Cheung, "Mtfh: A matrix tri-factorization hashing framework for efficient cross-modal retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 3, pp. 964–981, Sep. 2019. [Online]. Available: https://doi.org/10.1109/TPAMI.2019.2940446

[14] C. Bai, C. Zeng, Q. Ma, J. Zhang, and S. Chen, "Deep adversarial discrete hashing for cross-modal retrieval," *ICMR '20: Proceedings of the 2020 International Conference on Multimedia Retrieval*, p. 525–531, Jun. 2020. [Online]. Available: https://doi.org/10.1145/3372278.3390711

[15] H. Zhao, X. She, S. Wang, and K. Ma, "Fast discrete matrix factorization hashing for large-scale cross-modal retrieval," *International Conference on Multimedia Modeling. Springer, Cham*, pp. 24–36, Jan. 2021. [Online]. Available: https://doi.org/10.1109/TCYB.2015.2392052

[16] P. Indyk, R. Motwani, P. Raghavan, and S. S. Vempala, "Locality-preserving hashing in multidimensional spaces," in *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, F. T. Leighton and P. W. Shor, Eds. ACM, 1997, pp. 618–625. [Online]. Available: https://doi.org/10.1145/258533.258656

[17] L. Yuan, T. Wang, X. Zhang, F. E. Tay, Z. Jie, W. Liu, and J. Feng, "Central similarity quantization for efficient image and video retrieval," *In Proceedings of of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, pp. 3083–3092, 2020. [Online]. Available: https://doi.org/10.1109/CVPR42600.2020.00315

[18] T. van Erven, P. Harremos, and J. Shawe-Taylor, "Rényi divergence and kullback-leibler divergence," *IEEE Transactions on Information Theory*, vol. 60, no. 7, pp. 3797 – 3820, Jul. 2014. [Online]. Available: https://doi.org/10.1109/TIT.2014.2320500

[19] K. Simonyan and A. Zisserman. (2015, Feb.) Very deep convolutional networks for large-scale image recognition. [Online]. Available: https://arxiv.org/abs/1409.1556

[20] L. van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. 85, pp. 2579–2605, Aug. 2008.

[21] C. H. Papadimitriou, "On the complexity of integer programming," *J. ACM*, vol. 28, no. 4, Oct. 1981. [Online]. Available: https://doi.org/10.1145/322276.322287

[22] S. Ruder. (2017, Jul.) An overview of gradient descent optimization algorithms. [Online]. Available: https://arxiv.org/abs/1609.04747

[23] M. E. Tipping, "Sparse bayesian learning and the relevance vector machine," *Journal of Machine Learning Research*, vol. 1, no. 3, pp. 211–244, Jan. 2001. [Online]. Available: https://doi.org/10.1162/15324430152748236

[24] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, p. 2825–2830, 2011. [Online]. Available: https://hal.inria.fr/hal-00650905v2

[25] M. J. Huiskes and M. Lew, "The mir flickr retrieval evaluation," *Proceedings of the 1st ACM international conference on Multimedia information retrieval*, pp. 39–43, Oct. 2008. [Online]. Available: http://doi.acm.org/10.1145/1460096.1460104

[26] G. Ding, Y. Guo, and J. Zhou, "Collective matrix factorization hashing for multimodal data," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2075–2082, Jun. 2014. [Online]. Available: https://doi.org/10.1109/CVPR.2014.267

[27] V. Garcia, E. Debreuve, and M. Barlaud, "Fast k nearest neighbor search using gpu," *Proceedings of the 2019 on International Conference on Multimedia Retrieval*, Jun. 2008. [Online]. Available: https://doi.org/10.1109/CVPRW.2008.4563100

[28] B. Derrick, D. Toher, and P. White, "How to compare the means of two samples that include paired observations and independent observations: A companion to derrick, russ, toher and white (2017)," *The Quantitative Methods for Psychology*, vol. 13, no. 2, pp. 120–126, Apr. 2017. [Online]. Available: https://doi.org/10.20982/tqmp.13.2.p120