Deep image hashing based on twin-bottleneck hashing with variational autoencoders

Maxim Verwilst*, Nina Žižakić*, Lingchen Gu†* and Aleksandra Pižurica*
* *Group for Artificial Intelligence and Sparse Modelling (GAIM), TELIN, Ghent University*, Ghent, Belgium
{maxim.verwilst, nina.zizakic, aleksandra.pizurica}@ugent.be
† *School of Information Science and Engineering, Shandong University*, Qingdao, China
lingchengu@mail.sdu.edu.cn

***Abstract*—With the ever-increasing availability of data, the need for efficient and accurate image retrieval methods has become larger and larger. Deep hashing has proven to be a promising solution, by defining a hash function to convert the data into a manageable lower-dimensional representation. In this paper, we apply recent insights from the field of variational autoencoders to the field of deep image hashing, thus achieving an improvement over the current state of the art as shown by experimental evaluation. The code used in this paper is open-source and available on GitHub[1].**

***Index Terms*—Image hashing, deep hashing, content-based image retrieval, variational autoencoders, unsupervised deep learning.**

I. Introduction

Due to the growth of computer vision applications, there has been an ever-increasing supply of high-dimensional visual data. Thus, there is an increase in demand for accurate and efficient retrieval methods for these datasets. Traditional nearest neighbour search methods tend to struggle with high-dimensional data. Not only are these techniques impacted by the increasing computational cost for calculating element-wise distances, they also fail to translate semantic information into these distances. A promising solution to this problem is presented in the field of image hashing. Here, data in high-dimensional space is mapped onto low-dimensional hash codes in such a way that distance between data points is conserved into those hash codes. This allows for the application of traditional methods on the converted low-dimensional data.

Hashing methods can be both data-independent and data-dependent (learning to hash). Data-independent hashing are based on hand-crafted features, and can thus only capture visual similarity between images, rather than a semantic similarity. Data-dependent methods, on the other hand, use deep learning to achieve semantic similarity and have shown state-of-the-art performance over classical data-independent methods [1]–[4]. Data-dependent hashing techniques can be subdivided in two classes: supervised and unsupervised methods. While supervised techniques can use labels to increase retrieval accuracy significantly, they also depend on said labels. In many real-world applications, datasets may not have semantic annotations. Here we opt for unsupervised hashing techniques, which use the intrinsic data structure to determine meaningful hash codes.

Many unsupervised techniques are already established, examples of which are autoencoder networks [3], [5], [6], adversarial networks [1], [7], [8] and graph-based networks [9], [10]. A recent work by Shen et al. [4] shows state-of-the-art results. The authors describe an autoencoder and graph-based hybrid network for hashing, called Twin Bottleneck Hashing (TBH). The architecture leverages a code-driven graph, allowing it to circumvent the static-graph problem which is inherent with precomputed graphs.

In this work, we leverage and optimize the architecture from twin bottleneck hashing with recent insights from the field of variational autoencoders. In particular, we propose two separate improvements over TBH and experimentally show an overall increased performance on the CIFAR-10 dataset for both improvements separately, as well as together.

The rest of the paper is organised as follows. We provide an overview of state-of-the-art on hashing for content-based image retrieval in Section II. In Section III, we present our image hashing method which builds upon TBH by improving the binary bottleneck (Section III-A) and expanding the continuous bottleneck (Section III-B). In Section IV, we experimentally evaluate the two modifications, both separately and together, in all cases showing improvement over the state of the art. We conclude our work in Section V.

II. Related work

A. Hashing for content-based image retrieval

Hashing techniques can be categorised into data-independent and data-dependent methods (learning to hash). Data-independent methods tend to rely on designed features to extract hash codes, without using the data distribution. These methods are often referred to as locality sensitive hashing methods, named after a family of hash functions that maps similar inputs to the same hash code. Two works pioneered this technique [11], [12]. There are a lot of variations using different distance metrics [12]–[16] or changing the search method [17]–[19].

Data-independent methods may not optimally exploit the full data distribution. These methods often fail to capture semantic similarities between data points [20]. Data-driven methods are showing much more promising results in the

[1] https://github.com/maximverwilst/deepimagehashing-VAE

field of image retrieval [21]. Most of the work in this field has focused so far on supervised techniques, which exploit the data distribution together with its annotations. Over recent years, many different architectures have been proposed for supervised learning to hash. Perhaps the most straightforward techniques make use of a deep encoder to directly encode inputs to hash codes [2], [22]–[24]. Yang et al. presented a fine-tuning process to convert such an encoder of a classifier into a deep hashing network [25]. Another popular supervised technique trains on predicting a pairwise loss function [26]–[29]. A natural extension to this are triplet-based losses, where a query data point is compared to both a similar and a dissimilar data point [30]. Supervised generative adversarial networks have proven to be viable hashing methods [31]–[33], as well as supervised autoencoder-based networks that exploit the models' bottleneck to extract hash codes [34], [35].

While supervised data-driven methods do show promising retrieval results, they require annotated datasets. This is a severe limitation as for many real-world datasets, it is not feasible or it is simply too costly to annotate. Therefore, there is a demand for unsupervised methods that can learn useful hash functions solely based on the data distribution. There is a wide variety in the type of unsupervised methods. Generative adversarial networks again show state-of-the-art performance in this field [1], [7], [8], [31]. Similarly, autoencoders also remain a relevant technique [3], [5], [36]. Dai et al. build on the idea of a variational autoencoder, applying it to construct hash codes directly from the bottleneck [6]. Hu et al. propose an unsupervised technique which assigns pseudo labels to the data using precomputed features and shows promising results [37]. The approach optimises its hash function to maximally compress the dataset and is a generative approach since it can be used to regenerate the inputs. A recent work by Shen et al. [4] introduced an autoencoder-type architecture which implements elements from graph-based learning called auto-encoding twin-bottleneck hashing (TBH). This architecture inspired our work and thus we describe it in more detail in Section II-C. While TBH provides state-of-the-art unsupervised hashing, we show that by improving certain components of the architecture we can improve its performance even further.

B. Variational autoencoders

Variational autoencoders were first proposed by Kingma et al. [38] as a probabilistic version of the classical autoencoders. Since then, variational autoencoders have seen improvements and adaptations in the last decade, we will briefly touch on the developments that partly inspired this work. Higgins et al. [39] adapt the standard variational autoencoder to include interpretable and factorised latent representations. This work is expanded upon by Chen et al. [40]. Here a novel loss function is introduced without the need for additional hyper-parameters. The work by Rezende et al. includes an algorithm for constrained optimisation [41]. This allows for a robust way to balance reconstruction and compression constraints.

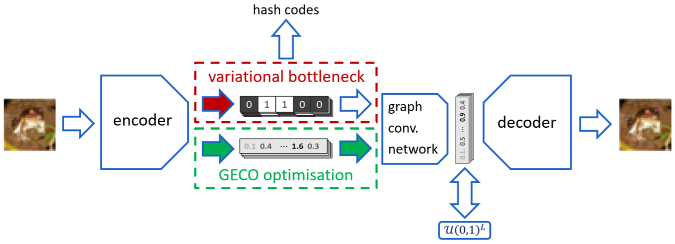

Fig. 1. Schematic of the TBH architecture with proposed improvements. In the binary bottleneck (top bottleneck) we change the generation of hash codes to be based on variational autoencoder with disentangled variables and we omit the regulariser (Section III-A). We also change the continuous bottleneck (bottom bottleneck) to use a variational autoencoder that is trained using a constrained optimisation setup, in order to better control the trade-off between compression and reconstruction quality of generated samples (Section III-B).

C. Twin bottleneck hashing

The network structure of TBH is an adapted autoencoder architecture, the main components of which are explained here briefly.

1) Twin bottlenecks: The most notable feature of TBH is the use of two separate bottlenecks instead of one. First, a binary bottleneck, which is responsible for generating the actual hash codes. Inspired by Dai et al., a stochastic neuron is used to generate the hash codes [6]. An intrinsic problem that arises is that we want binary codes to remain as small as possible, this limits the flexibility of the model. To alleviate this problem, a second larger continuous bottleneck allows for the model to capture much more complex representations.

2) Graph convolutional layer: The graph convolutional layer creates a pathway to calculate the gradients for both bottlenecks. It uses the binary variables to create a similarity graph, which is used to create the convolutional filter along with a set of trainable weights. Note that intuitively, the graph convolutional layer will penalise unrelated samples that are closer together in Hamming space. One of the major contributions of TBH is that the similarity graph is now constructed and optimised during training, rather than built on precomputed features.

3) Regularisation: TBH uses two discriminators as a means of regularisation. One discriminates hash codes from random binary codes. The other one discriminates the output of the graph convolutional layer with uniformly distributed samples between zero and one.

III. Proposed Method

Here we propose an unsupervised hashing approach, which builds on the twin bottleneck hashing (TBH) model and improves upon it. Specifically we improve two main aspects of the TBH approach. First, we change the way binary codes are generated. This way we directly influence the ability of the model to learn representations. Second, we design a more powerful generative bottleneck. This improves the model as a whole, and thus also the retrieval scores. Our overall objective function consists of three components: a regularisation term on the binary bottleneck L_{BBN}, a constrained optimisation setup between the reconstruction loss and the regularisation on the continuous bottleneck L_{CBN} and lastly a discriminator term on the output of the graph convolutional network L_D analogous to TBH's approach. This results in the following objective:

$$L_{OBJ} = L_{BBN} + L_{CBN} + L_D\,, \tag{1}$$

with L_{BBN} and L_{CBN} explained in more detail in the following sections.

A. Improving the binary bottleneck

We focus first on the generation of the binary hash codes. Instead of generating hash codes with a stochastic neuron [6] like in TBH, we use a variational autoencoder scheme. In recent years, variational-autoencoder–based methods have shown improvements over the state of the art in image generation, classification and particularly representation learning. This motivates us to explore their potential for improved image hashing too. Learning a better representation in the bottleneck will enable us to capture more information from the input images to the hash codes and thus achieve better retrieval accuracies.

We thus design the binary bottleneck, which generates hash codes, as a variational autoencoder with stochastic sampling. We intend to optimise the hash codes to have statistically independent bits. This relates to a common topic in variational autoencoders where the latent distribution in the bottleneck is optimised to be disentangled. Chen et al. [40] stated that Kullback-Leibler (KL) divergence term in the ELBO objective for variational autoencoders can be decomposed in three terms: Index-Code mutual information, total correlation and dimension-wise KL divergence. They claimed that penalising the total correlation term by an extra factor would be responsible for a disentangled latent distribution. This results in latent variables that map to more explainable features like colour, shape etc., but it sacrifices some quality of the models' reconstructions.

We apply these principles to our hash code generation to measure if disentangled representations would benefit our models' performance. We formulate the loss on the binary bottleneck L_{BBN} as in a standard variational autoencoder with the decomposed KL term.

$$\begin{aligned} L_{BBN} = KL[q(z,n)||q(z)p(n)] + \\ \beta * KL[q(z)||\prod_j q(z_j)] + \sum_j KL[q(z_j)||p(z_j)]\,, \end{aligned} \tag{2}$$

where β is a new hyper-parameter to penalise the total correlation term, x and z are the data and latent variable respectively, and n is a uniform random variable on $\{1, 2, ..., N\}$ which indexes the data points. Our encoder is defined as $q(z|n) = q(z|x_n)$ and the data's prior distribution $p(n) = 1/N$. From these we can derive $q(z, n) = q(z|n)p(n) = q(z|n)\frac{1}{N}$ and the aggregated posterior $q(z) = \sum_{n=1}^{N} q(z|n)p(n)$. Following [40], we express $KL[q(z, n)||q(z)p(n)]$ as the index-code mutual information. This is the mutual information between the data variable n and the latent variable z on the empirical data distribution $q(z, n)$. The second term, $\beta * KL[q(z)|| \prod_j q(z_j)]$, is referred to as the total correlation. The penalty on this term forces the model to find statistically independent factors in the data distribution. The last term, $\sum_j KL[q(z_j)||p(z_j)]$, is the dimension-wise KL divergence, which encourages individual latent dimensions to represent their corresponding prior.

We expect that hashing with disentangled latent variables will force bits to allocate to clear, explainable features, allowing our model to preserve semantic similarity between samples. Note that these extra constraints on the bottleneck could also worsen the models' ability to fit to the input data. Experimentally however, we see a slight improvement in the precision of the architecture as will be shown in Section IV. In addition, we omit the discriminator on the binary bottleneck, as our implementation now uses KL divergence to regularise the binary bottleneck.

B. Expanding the continuous bottleneck

Now we turn to the design of the continuous bottleneck. Our idea is to convert the more standard autoencoder structure to a variational-autoencoder–type structure. This expanded architecture should allow the model to better capture input data, specifically boosting the shared encoder and thus achieve better results when generating hash codes. Rezende et al. [41] propose a robust algorithm for optimising variational autoencoders. We apply their method on the continuous bottleneck and redefine accordingly the ELBO objective into a constraint optimisation problem:

$$\begin{aligned} L_{CBN} = \mathbb{E}_{p(x)}[KL[q(z|x); \pi(z)]] \\ + \lambda^T \mathbb{E}_{p(x)q(z|x)}[C(x, g(z))]\,, \end{aligned} \tag{3}$$

$$\text{with } C(x, g(z)) = \|x - g(z)\|^2 - \kappa\,, \tag{4}$$

where we balance compression $\mathbb{E}_{p(x)}[KL[q(z|x); \pi(z)]]$ on the bottleneck with reconstruction quality of our generated samples $\mathbb{E}_{p(x)q(z|x)}[C(x, g(z))]$. For the compression term, we use the KL divergence between the encoder's generated Gaussian distributions $q(z|x)$ and a set of normalised Gaussian priors $\pi(z)$. We use L2-loss, $\|x - g(z)\|^2$, to represent our reconstruction loss. This allows for an elegant trade-off where we can choose to invest a bit more in compression as we do not care much about our generated samples. A new hyper-parameter κ is set to achieve the desired reconstruction quality, which controls the value of λ^T, as is described in the optimisation scheme proposed by Rezende et al. [41].

We expect as the model improves through the continuous bottleneck, that the shared encoder and decoder will improve. This benefits the binary bottleneck directly and should result in better retrieval scores for our model. In the following section, we will show that this indeed improves the performance of the model.

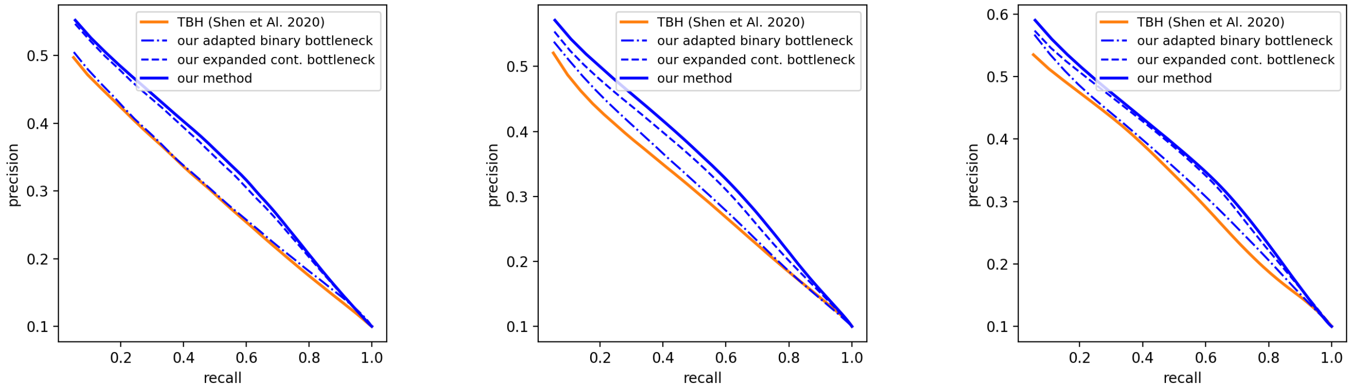

Fig. 2. Comparison of precision-recall curves for both TBH, our method and improvements from sections III-A and III-B separately on the CIFAR-10 dataset. We use 16-bit, 32-bit and 64-bit codes (left to right respectively) to evaluate these models.

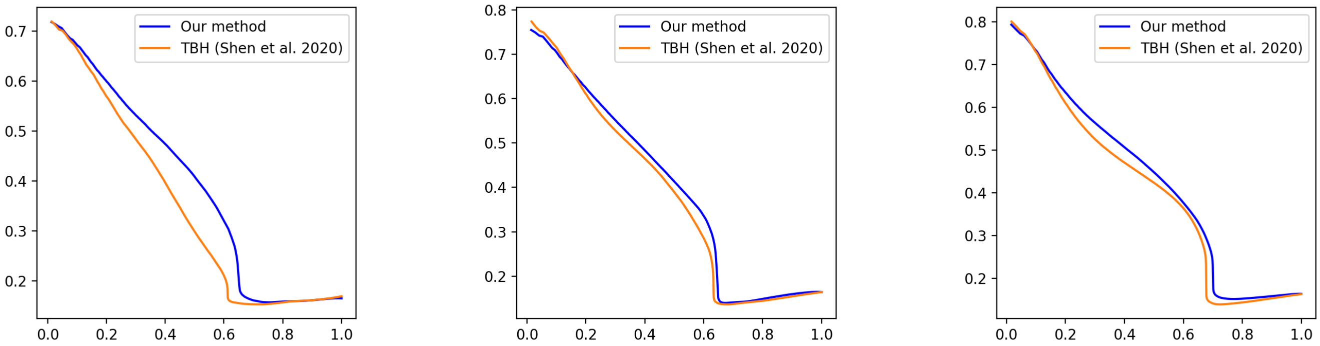

Fig. 3. Comparison of precision-recall curves for TBH and our method on the MS-COCO dataset. We use 16-bit, 32-bit and 64-bit codes (left to right respectively) to evaluate the models.

IV. Experimental results

We evaluate our method in comparison with the related state-of-the-art methods in the field of unsupervised hashing. We perform experiments on both CIFAR-10 and MS-COCO dataset. Additionally, we extract features from the fc_7 layer using AlexNet [42]. This is the same extraction method as described in TBH. We train and evaluate our models in TensorFlow. For the baseline method TBH, we experiment with the same setup and hyper-parameters as stated in their paper. As for our method, we use the following hyper-parameters. TBH hyper-parameters are kept the same, with $\lambda = 1$ and $L = 512$. We train in batches with batch size $= 1024$ using learning rate $= 1e-4$. For the GECO implementation we use $\kappa = 2400$, and clip the λ parameter between $1e-6$ and $1e12$ to avoid extreme values during training.

Figure 2 shows precision-recall curves for our method, the TBH baseline and each of the two improvements from sections III-A and III-B separately on CIFAR-10 dataset. We see that our method improves precision scores across the board for 16, 32 and 64 bits codes. This confirms that the changes we make to the bottlenecks of TBH do in fact improve the generated hash codes and thus the retrieval scores. Looking at the components separately, we see that expanding the continuous bottleneck contributes to the majority of our improvement. Changing the binary bottleneck has a less significant effect.

Figure 3 shows precision-recall curves for our method and TBH baseline on MS-COCO dataset. On this dataset, the curves for both methods look different in comparison to those on the CIFAR-10 dataset. This suggests that there is a group of data samples that cause confusion and retrieving later samples starts to increase precision. Our method outperforms TBH on this dataset as well.

Table I shows a comparison between more state-of-the-art methods on CIFAR-10 dataset. To evaluate methods, we use mean average precision (mAP) at 1000. This measures the average relevance scores of a set of the top-1000 images in response to a query. Our method shows improved results over multiple baselines.

To conclude the results, we show some example retrievals from both our method and the TBH baseline in Figure 4. The query examples are randomly selected from the test set and retrievals are returned from our train set. On such a small set of queries, it is difficult to clearly see our improvements – there are examples of images where our method performs

TABLE I
mAP of our method and state-of-the-art unsupervised hashing methods on the CIFAR-10 dataset.

Method	16 bits	32 bits	64 bits
LSH [11]	0.106	0.102	0.105
SpH [9]	0.272	0.285	0.300
AGH [43]	0.333	0.357	0.358
SpherH [44]	0.254	0.291	0.333
KMH [45]	0.279	0.296	0.334
ITQ [23]	0.305	0.325	0.349
DGH [46]	0.335	0.353	0.361
DeepBit [47]	0.194	0.249	0.277
SGH [6]	0.435	0.437	0.433
BGAN [8]	0.525	0.531	0.562
BinGAN [7]	0.476	0.5122	0.520
GreedyHash [48]	0.448	0.473	0.501
HashGAN [31]	0.447	0.463	0.481
DVB [49]	0.403	0.422	0.446
DistillHash [50]	0.284	0.285	0.288
PseudoLabel [37]	0.517	0.572	0.596
TBH [4]	0.532	0.573	0.578
Proposed method	**0.556**	**0.6021**	**0.6057**

slightly better than TBH, and vice versa. For the most part, however, we observe that queries where our method struggles are the same ones that are difficult TBH method, which is to be expected since these are more difficult queries and since these two methods share similarities.

V. Conclusion

In this paper, an improved method for unsupervised hashing based on twin bottleneck hashing is proposed. We improve the performance of the state-of-the-art TBH approach by designing the bottleneck structure inspired by some recent insights in the field of variational autoencoders. We adapt the generation of hash codes to promote learning disentangled representations and expand the continuous bottleneck to a variational autoencoder to improve the models' ability to fit to input data. The experiments show that the proposed architecture has improved retrieval performance when compared to state-of-the-art methods.

Acknowledgements

This research received funding from the Flemish Government (AI Research Program).

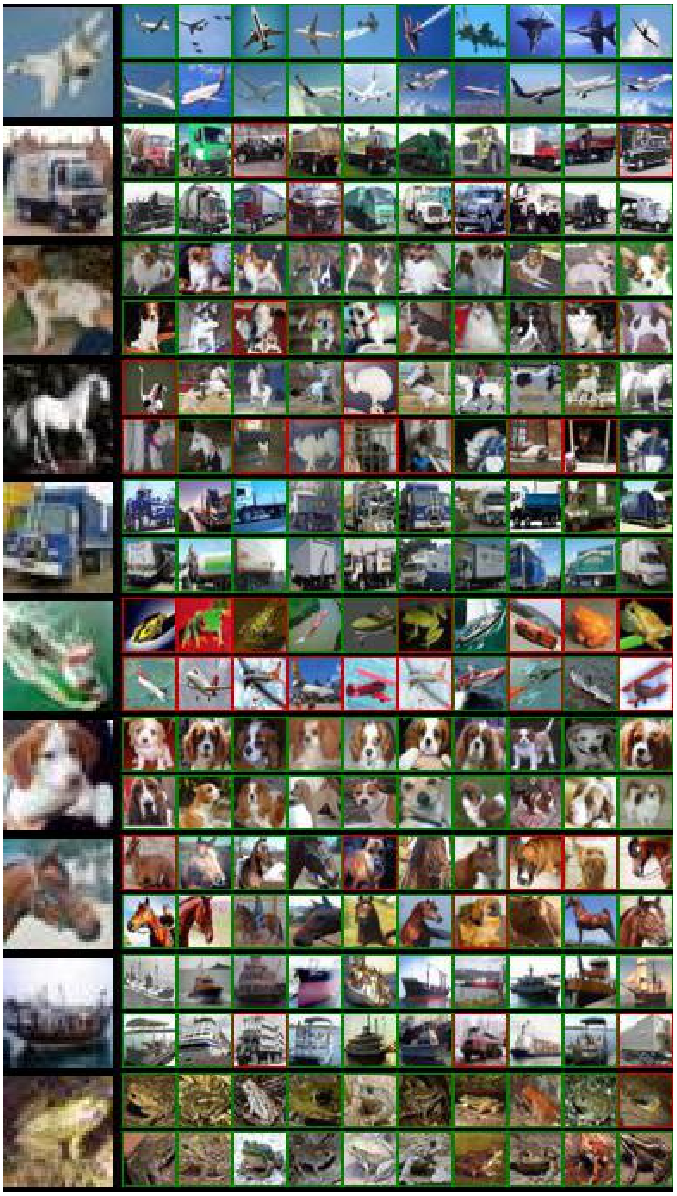

Fig. 4. Retrieval examples for both TBH (bottom row) [4] and our method (top row), the correct classes are labeled green.

References

[1] K. G. Dizaji, F. Zheng, N. Sadoughi, Y. Yang, C. Deng, and H. Huang, "Unsupervised deep generative adversarial hashing network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 3664–3673.

[2] F. Shen, C. Shen, W. Liu, and H. Tao Shen, "Supervised discrete hashing," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 37–45.

[3] R. R. Miguel A. Carreira-Perpinan, "Deep Hashing for Compact Binary Codes Learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 557–566.

[4] Y. Shen, J. Qin, J. Chen, M. Yu, L. Liu, F. Zhu, F. Shen, and L. Shao, "Auto-Encoding Twin-Bottleneck Hashing," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2815–2824, 2020.

[5] S. En, B. Crémilleux, and F. Jurie, "Unsupervised deep hashing with stacked convolutional autoencoders," in *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2017, pp. 3420–3424.

[6] B. Dai, R. Guo, S. Kumar, N. He, and L. Song, "Stochastic generative hashing," in *International Conference on Machine Learning (ICML)*. PMLR, 2017, pp. 913–922.

[7] M. Zieba, P. Semberecki, T. El-Gaaly, and T. Trzcinski, "BinGAN: Learning Compact Binary Descriptors with a Regularized GAN," *arXiv preprint arXiv:1806.06778*, 2018.

[8] J. Song, T. He, L. Gao, X. Xu, A. Hanjalic, and H. T. Shen, "Binary Generative Adversarial Networks for Image Retrieval," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr. 2018.

[9] Y. Weiss, A. Torralba, and R. Fergus, "Spectral Hashing," in *Advances in Neural Information Processing Systems 21*. Curran Associates, Inc., 2009, pp. 1753–1760.

[10] W. Liu, J. Wang, S. Kumar, and S.-F. Chang, "Hashing with graphs," in *International Conference on Machine Learning (ICML)*, 2011.

[11] A. Andoni and P. Indyk, "Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions," in *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, 2006, pp. 459–468.

[12] M. S. Charikar, "Similarity estimation techniques from rounding algorithms," in *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, 2002, pp. 380–388.

[13] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," in *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, 1998, pp. 604–613.

[14] A. Dasgupta, R. Kumar, and T. Sarlós, "Fast locality-sensitive hashing," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2011, pp. 1073–1081.

[15] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 6, pp. 1092–1104, 2011.

[16] J. Ji, J. Li, S. Yan, B. Zhang, and Q. Tian, "Super-bit locality-sensitive hashing," in *Advances in Neural Information Processing Systems*. Citeseer, 2012, pp. 108–116.

[17] R. Panigrahy, "Entropy based nearest neighbor search in high dimensions," *arXiv preprint cs/0510019*, 2005.

[18] M. Bawa, T. Condie, and P. Ganesan, "LSH forest: self-tuning indexes for similarity search," in *Proceedings of the 14th international conference on World Wide Web*, 2005, pp. 651–660.

[19] J. Pan and D. Manocha, "Bi-level locality sensitive hashing for k-nearest neighbor computation," in *2012 IEEE 28th International Conference on Data Engineering*, 2012, pp. 378–389.

[20] J. Wang, T. Zhang, N. Sebe, H. T. Shen *et al.*, "A survey on learning to hash," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 769–790, 2017.

[21] X. Luo, C. Chen, H. Zhong, H. Zhang, M. Deng, J. Huang, and X. Hua, "A survey on deep hashing methods," *arXiv preprint arXiv:2003.03369*, 2020.

[22] V. Erin Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou, "Deep Hashing for Compact Binary Codes Learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[23] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin, "Iterative Quantization: A Procrustean Approach to Learning Binary Codes for Large-Scale Image Retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 12, pp. 2916–2929, 2013.

[24] X. Wang, T. Zhang, G.-J. Qi, J. Tang, and J. Wang, "Supervised quantization for similarity search," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2018–2026.

[25] H.-F. Yang, K. Lin, and C.-S. Chen, "Supervised learning of semantics-preserving hash via deep convolutional neural networks," *IEEE transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 2, pp. 437–451, 2017.

[26] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan, "Supervised hashing for image retrieval via image representation learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 28, no. 1, 2014.

[27] H. Liu, R. Wang, S. Shan, and X. Chen, "Deep supervised hashing for fast image retrieval," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2064–2072.

[28] Y. Lin, R. Jin, D. Cai, S. Yan, and X. Li, "Compressed hashing," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 446–451.

[29] X. Shi, F. Xing, J. Cai, Z. Zhang, Y. Xie, and L. Yang, "Kernel-based supervised discrete hashing for image retrieval," in *European Conference on Computer Vision (ECCV)*. Springer, 2016, pp. 419–433.

[30] X. Wang, Y. Shi, and K. M. Kitani, "Deep supervised hashing with triplet labels," in *Asian conference on computer vision*. Springer, 2016, pp. 70–84.

[31] Y. Cao, B. Liu, M. Long, J. Wang, and M. KLiss, "HashGAN: Deep Learning to Hash with Pair Conditional Wasserstein GAN," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 1287–1296.

[32] Z. Qiu, Y. Pan, T. Yao, and T. Mei, "Deep semantic hashing with generative adversarial networks," in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2017, pp. 225–234.

[33] G. Wang, Q. Hu, J. Cheng, and Z. Hou, "Semi-supervised generative adversarial hashing for image retrieval," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 469–485.

[34] S. Z. Dadaneh, S. Boluki, M. Yin, M. Zhou, and X. Qian, "Pairwise supervised hashing with Bernoulli variational auto-encoder and self-control gradient estimator," in *Conference on Uncertainty in Artificial Intelligence*. PMLR, 2020, pp. 540–549.

[35] Y. Cao, M. Long, J. Wang, and H. Zhu, "Correlation autoencoder hashing for supervised cross-modal search," in *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*, 2016, pp. 197–204.

[36] C. Hansen, C. Hansen, J. G. Simonsen, S. Alstrup, and C. Lioma, "Unsupervised Semantic Hashing with Pairwise Reconstruction," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 2009–2012.

[37] Q. Hu, J. Wu, J. Cheng, L. Wu, and H. Lu, "Pseudo Label Based Unsupervised Deep Discriminative Hashing for Image Retrieval," in *Proceedings of the 25th ACM International Conference on Multimedia*, ser. MM '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 1584–1590.

[38] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[39] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "β-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework," in *International Conference on Learning Representations (ICLR)*, 2017.

[40] R. T. Chen, X. Li, R. Grosse, and D. Duvenaud, "Isolating Sources of Disentanglement in Variational Autoencoders," *arXiv preprint arXiv:1802.04942*, 2018.

[41] D. J. Rezende and F. Viola, "Taming VAEs," *arXiv preprint arXiv:1810.00597*, 2018.

[42] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," *Technical report, University of Toronto*, 2009.

[43] W. Liu, J. Wang, S. Kumar, and S.-F. Chang, "Hashing with graphs," in *International Conference on Machine Learning (ICML)*, 2011.

[44] J. Heo, Y. Lee, J. He, S. Chang, and S. Yoon, "Spherical hashing," in *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 2957–2964.

[45] K. He, F. Wen, and J. Sun, "K-Means Hashing: An Affinity-Preserving Quantization Method for Learning Binary Compact Codes," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2938–2945.

[46] W. Liu, C. Mu, S. Kumar, and S.-F. Chang, "Discrete Graph Hashing," in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, Eds., vol. 27. Curran Associates, Inc., 2014.

[47] K. Lin, J. Lu, C.-S. Chen, and J. Zhou, "Learning Compact Binary Descriptors With Unsupervised Deep Neural Networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[48] S. Su, C. Zhang, K. Han, and Y. Tian, "Greedy Hash: Towards Fast Optimization for Accurate Hash Coding in CNN," in *Advances in Neural Information Processing Systems*, vol. 31. Curran Associates, Inc., 2018.

[49] Y. Shen, L. Liu, and L. Shao, "Unsupervised Binary Representation Learning with Deep Variational Networks," *International Journal of Computer Vision*, pp. 1–15, 2019.

[50] E. Yang, T. Liu, C. Deng, W. Liu, and D. Tao, "DistillHash: Unsupervised Deep Hashing by Distilling Data Pairs," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.