Supplemental Material for “Locally Linear Hashing for Extracting Non-Linear Manifolds” – Additional Experimental Results –

Go Irie
NTT Corporation
Kanagawa, Japan
irie.go@lab.ntt.co.jp

Zhenguo Li
Huawei Noah’s Ark Lab
Hong Kong, China
li.zhenguo@huawei.com

Xiao-Ming Wu, Shih-Fu Chang
Columbia University
New York, NY, USA
{xmwu, sfchang}@ee.columbia.edu

Abstract

This is the supplemental material for our main paper titled “Locally Linear Hashing for Extracting Non-Linear Manifolds”. In the main paper, we propose a new hashing method named Locally Linear Hashing (LLH) and report a set of extensive experimental results to demonstrate its superiority over the state-of-the-art methods. In this supplemental material, we provide additional results to emphasize its effectiveness, scalability, and efficiency.

Introduction

In our main paper, we experimentally compare our LLH to several state-of-the-art methods including ITQ [1], PQ-RR [4], SH [9], AGH [6], and IMH-tSNE [7], and demonstrate the significant superiority of our LLH over these methods (see Sec. 3 of our main paper). In this supplemental material, we show some additional results which cannot be presented in the main paper due to space limit.

Also, we compare our LLH to some other state-of-the-art methods including K-means Hashing (KMH) [2], Multidimensional Spectral Hashing (MDSH) [8], Spherical Hashing (SPH) [3], and Kernelized Locality Sensitive Hashing (KLSH) [5]. To run each method, we use the publicly available Matlab code provided by each author group with suggested parameter settings (if given) in each original paper. Specifically, for KMH which internally uses Product Quantization technique [4], the number of bits per sub-space is chosen from $\{2, 4, 8\}$ based on some preliminary tests[1]. For KLSH, we use the Gaussian RBF kernel $\kappa(x_i, x_j) = \exp(-||x_i - x_j||^2/2\sigma^2)$ and sample 300 training points to form the kernel Gramian. For KLSH and MDSH, the kernel (or affinity matrix) parameter σ is set to the average distance between data points in the training set.

[1]On the two synthetic datasets, Two-TrefoilKnots and Two-SwissRolls, the number of sub-spaces is set as 1 when $c = 1$, c being the number of bits.

For convenience, we hereafter call one set {**ITQ, PQ-RR, SH, AGH, IMH-tSNE**} **Group-A** and the other set {**KMH, MDSH, SPH, KLSH**} **Group-B**. For visibility, we show the comparative results with these two groups in separated figures.

For further details on experimental setup, protocols and datasets, please refer to our main paper.

Additional Results on Synthetic Datasets

Fig. 1 (Group-A) and Fig. 2 (Group-B) show the results on Two-TrefoilKnots. First, for every code length, our LLH outperforms all the methods in Group-B with significant performance gain (Fig. 2(a)). Note that our LLH also outperforms the methods in Group-A (see Fig. 6(b) in our main paper). Second, for each fixed {16, 32, 64}-bit code, our LLH shows significantly better performance than all the other methods at any number of top retrieved points (Fig. 1(b-d) and Fig. 2(b-d)). Notably our LLH obtains almost perfect precision in all the cases.

Fig. 3 (Group-A) and Fig. 4 (Group-B) show the results on Two-SwissRolls. For this dataset, no previous methods achieve reasonable performance, and only LLH (or LLH^0) works well and shows strong superiority to the other methods including the exhaustive l^2-scan. These results are clear evidences that our LLH can effectively capture and preserve the locally linear structures of the non-linear manifolds in the final Hamming space after hashing, which is indeed vital to the performance.

Additional Results on Face Images

We next show the results on Yale in Fig. 5 (Group-A) and Fig. 6 (Group-B). Note that the results for Group-A with 64-bit codes are presented in Fig. 8(b) in the main paper.

Our LLH is consistently better than all the other methods at every code length, and LLH^0 is the second best. The best competitive methods to ours are KMH and SPH, both aiming to preserve Euclidean neighborhood in the Hamming space. They outperform the exhaustive l^2-scan with relatively longer codes such as 24-bit or more (Fig. 6(a)). However, our LLH, and even LLH^0, are much better than these two methods at every code length and at every number of top retrieved points (Fig. 6(b-d)). The gain is also remarkable, ranging from 38% to 356%. These results suggest that our LLH successfully extracts the non-linear manifold structures on the Yale face dataset.

Additional Results on Handwritten Digits

Fig. 7 (Group-A) and Fig. 8 (Group-B) show the results on the USPS handwritten digits dataset. Our LLH shows the best performance on this dataset, and LLH^0 follows (Fig. 8(a)). While AGH is slightly better than LLH^0 in some cases (Fig. 7(a)), LLH is consistently better than AGH for all the cases.

Fig. 9 (Group-A) and Fig. 10 (Group-B) show the results on MNIST. Again, our LLH achieves the best performance in most cases. The best competitive method is AGH, and the performances of Group-B are relatively poor. For 16-bit (or less), LLH^0 performs better than LLH (Fig. 9(a)). This may be because 16 (or less) dimensional Hamming space is too low to preserve the locally linear structures of the multiple manifolds which is captrued by LLSR. In contrast, for 32-bits or more, LLH outperforms LLH^0 and AGH, especially when we retrieve 900 or less data points (Fig. 9(b)).

Additional Results on Natural Images

Fig. 11 (Group-A) and Fig. 12 (Group-B) show the results on CIFAR. Our LLH outperforms all the other methods in most cases. The best competitive method is ITQ, and SPH follows (Fig. 12(a)). Note that our LLH is the only method which is comparable to or even better than l^2-scan.

Fig. 13 (Group-A) and Fig. 14 (Group-B) show the results on ImageNet-200K. Our LLH is still the best on this dataset. LLH^0, KMH, SPH, and ITQ, are highly competitive on this dataset. With 64-bit codes, SPH is slightly better than our LLH when we retrieve more than 500 points (Fig. 14(d)). However, for shorter codes (from 16- to 48-bit codes), our LLH achieves higher precision (Fig. 14(a-c)) than SPH.

Lastly, Fig. 15 shows qualitative results on MIRFLICKR-1M. For every query, our LLH returns more semantically consistent images and shows better retrieval results than Group-B.

Computation Time

In this supplemental material, we focus on the scalability of the out-of-sample binary code generation time w.r.t. the database size n and K (see Sec. 2.3 in our main paper). All results are obtained on MIRFLICKR-1M dataset and with MATLAB on a workstation with 2.53 GHz Intel Xeon CPU and 64GB RAM.

Fig. 16(a) shows the binary code generation time for varied database sizes from $n = 50$K to 1M (in these experiments, we fix $m = 100$K). As analyzed in Sec. 2.4 in our main paper, the binary code generation time is independent of n. Fig. 16(b) shows the binary code generation time for varied K. We find that K does not really affect the code generation time in practice. This is because, as K increases, the number of data points within each cluster decreases (see Sec. 2.1 and Sec. 2.3), leading to almost constant time for code generation. Based on these results, our out-of-sample algorithm is scalable to both the database size n and K.

Parameter Study

We report some additional results on the parameter sensitivity of our LLH. Fig. 17(a-b) show the averaged precision on CIFAR with different settings of λ and η. Similar to the results on MNIST (repoted in Fig. 16 of our main paper), the performance is slightly sensitive to λ and η, but the retrieval quality is still much better than l^2-scan in most cases, especially when using longer codes such as 32-bit or more. Fig. 17(c-d) show the performance of LLH with varied K on CIFAR and MNIST datasets, respectively. We find that the performance of our LLH is quite stable w.r.t. K.

References

[1] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Trans. PAMI*, 35(12):2916–2929, 2013. 1

[2] K. He, F. Wen, and J. Sun. K-means hashing: an affinity-preserving quantization method for learning binary compact codes. In *CVPR*, 2013. 1

[3] J.-P. Heo, Y. Lee, J. He, S.-F. Chang, and S.-E. Yoon. Spherical hashing. In *CVPR*, 2012. 1

[4] H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE Trans. PAMI*, 33(1):117–128, 2011. 1

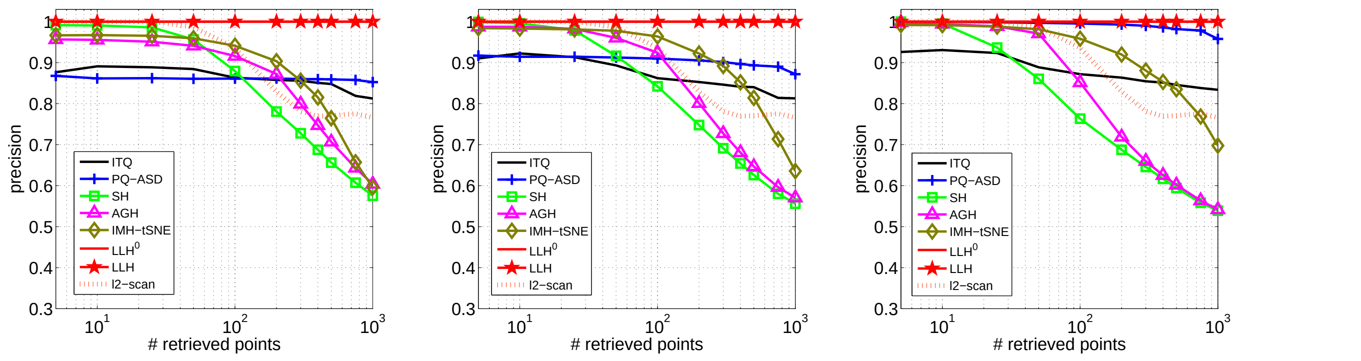
[5] B. Kulis, P. Jain, and K. Grauman. Kernelized locality-sensitive hashing. *IEEE Trans. PAMI*, 34(6):1092–1104, 2012. 1

[6] W. Liu, J. Wang, S. Kumar, and S.-F. Chang. Hashing with graphs. In *ICML*, 2011. 1

[7] F. Shen, C. Shen, Q. Shi, A. van den Hengel, and Z. Tang. Inductive hashing on manifolds. In *CVPR*, 2013. 1

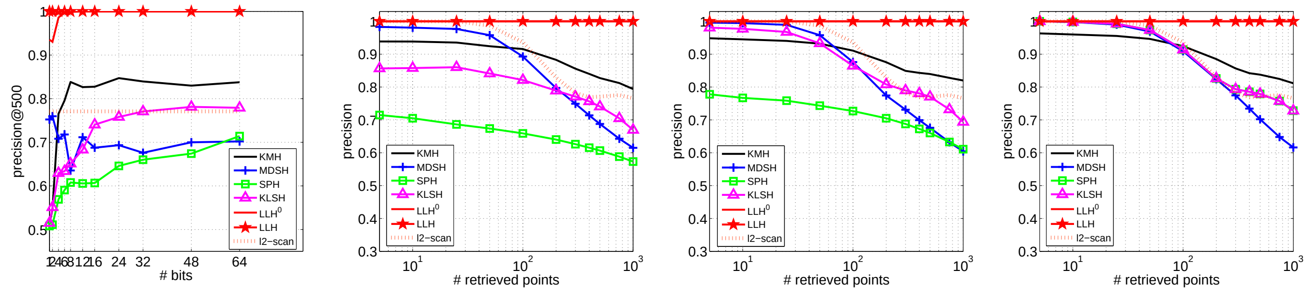
[8] Y. Weiss, R. Fergus, and A. Torralba. Multidimensional spectral hashing. In *ECCV*, 2012. 1

[9] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, 2008. 1



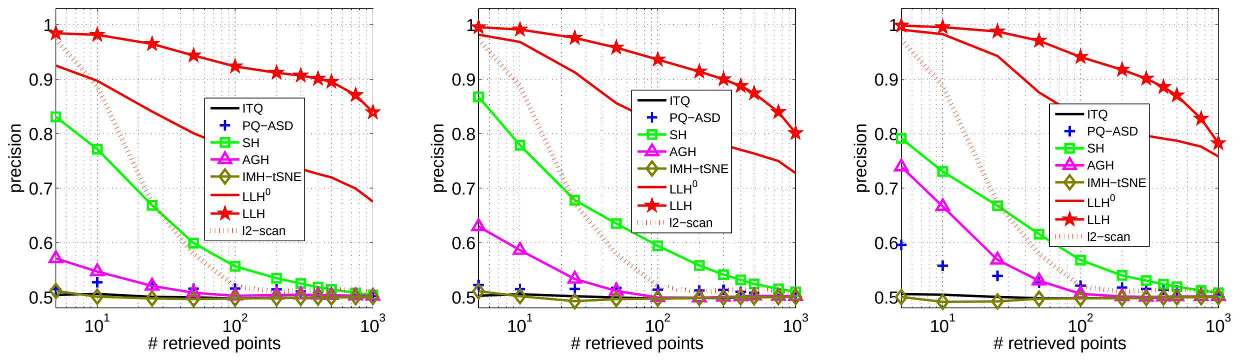
(a) (b) (c)

Figure 1. Comparison to Group-A on **Two-TrefoilKnots**. Averaged precision vs. the number of top retrieved points for (a) 16-bit, (b) 32-bit, and (c) 64-bit codes.



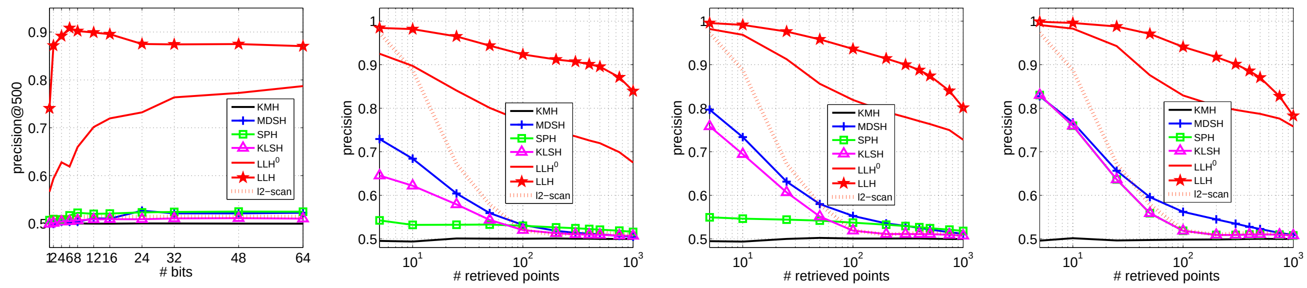
(a) (b) (c) (d)

Figure 2. Comparison to Group-B on **Two-TrefoilKnots**. (a) Averaged precision at top 500 retrieved points vs. the number of bits; Averaged precision vs. the number of top retrieved points for (b) 16-bit, (c) 32-bit, and (d) 64-bit codes.



(a) (b) (c)

Figure 3. Comparison to Group-A on **Two-SwissRolls**. Averaged precision vs. the number of top retrieved points for (a) 16-bit, (b) 32-bit, and (c) 64-bit codes.



(a) (b) (c) (d)

Figure 4. Comparison to Group-B on **Two-SwissRolls**. (a) Averaged precision at top 500 retrieved points vs. the number of bits; Averaged precision vs. the number of top retrieved points for (b) 16-bit, (c) 32-bit, and (d) 64-bit codes.

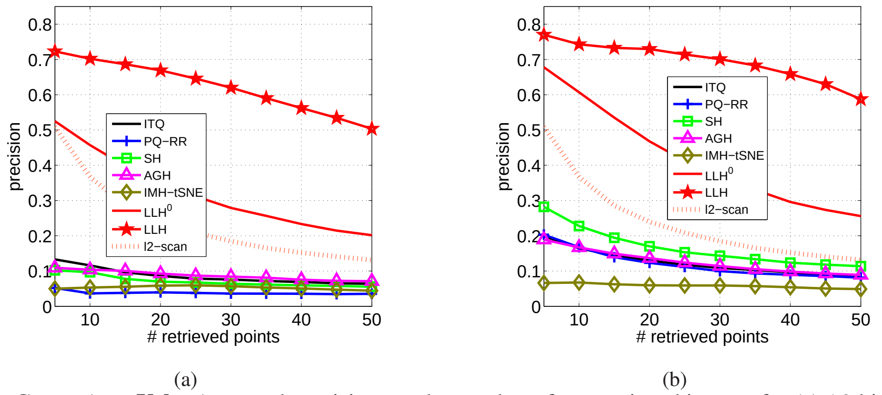


Figure 5. Comparison to Group-A on **Yale**. Averaged precision vs. the number of top retrieved images for (a) 16-bit and (b) 32-bit codes.

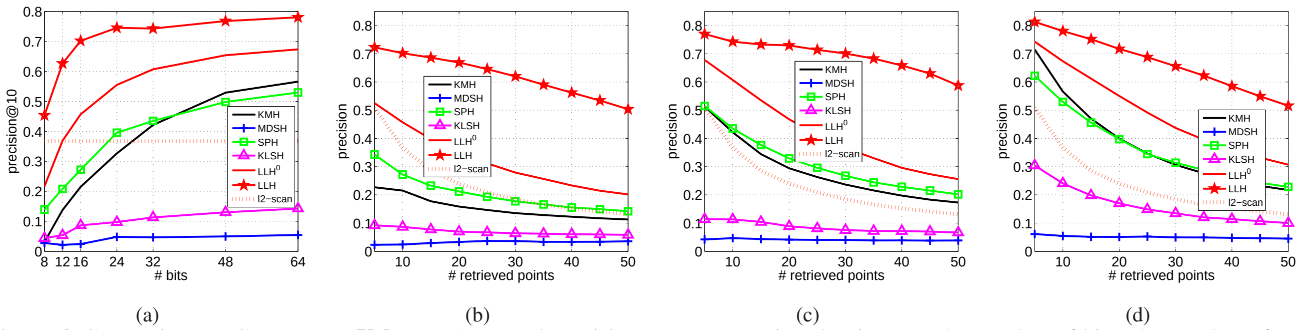


Figure 6. Comparison to Group-B on **Yale**. (a) Averaged precision at top 10 retrieved points vs. the number of bits; the number of top retrieved images for (b) 16-bit, (c) 32-bit, and (d) 64-bit codes.

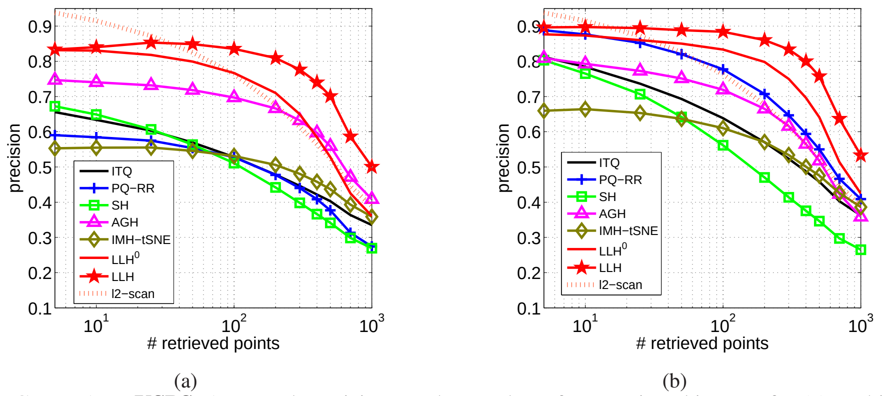


Figure 7. Comparison to Group-A on **USPS**; Averaged precision vs. the number of top retrieved images for (a) 16-bit and (b) 32-bit codes.

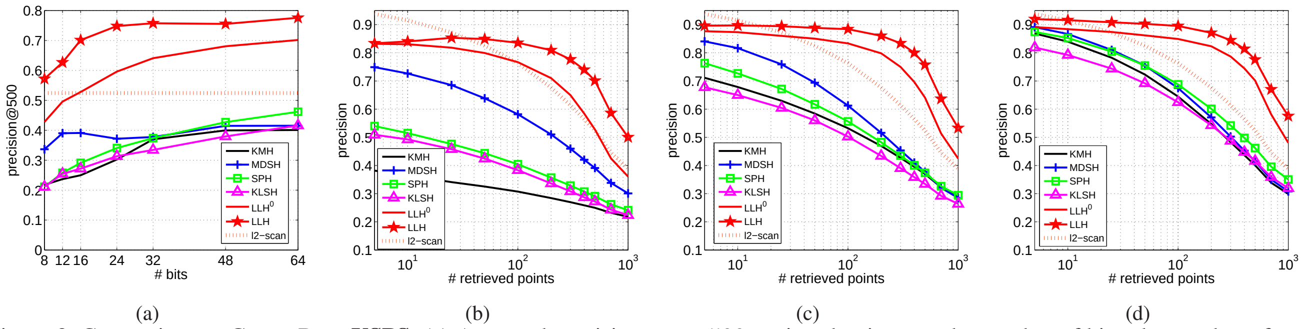
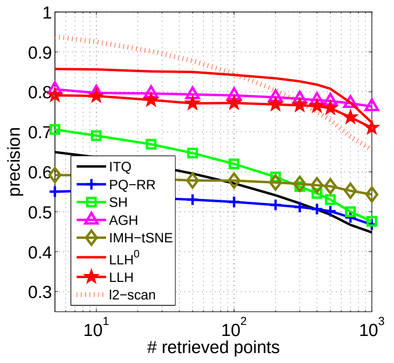


Figure 8. Comparison to Group-B on **USPS**; (a) Averaged precision at top 500 retrieved points vs. the number of bits; the number of top retrieved images for (b) 16-bit, (c) 32-bit, and (d) 64-bit codes.


(a)

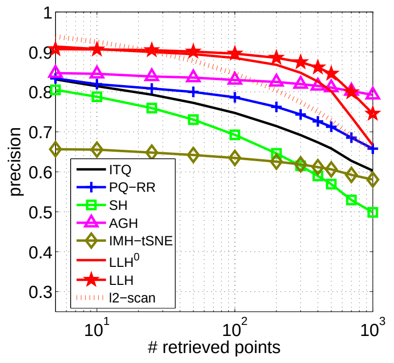
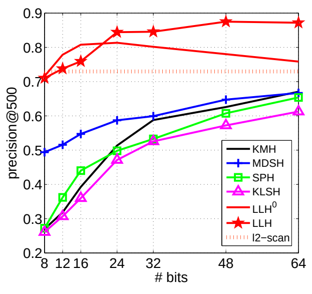
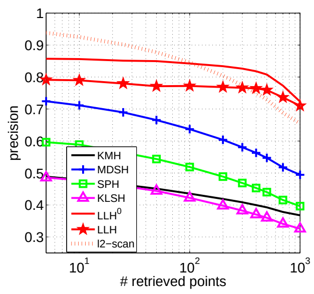
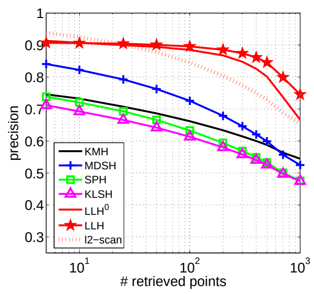

(b)

Figure 9. Comparison to Group-A on **MNIST**; Averaged precision vs. the number of top retrieved images for (a) 16-bit and (b) 32-bit codes.


(a)


(b)


(c)

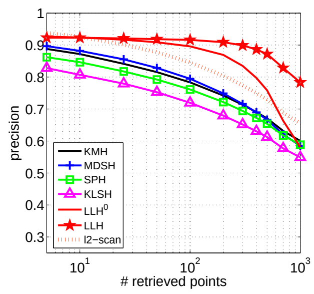
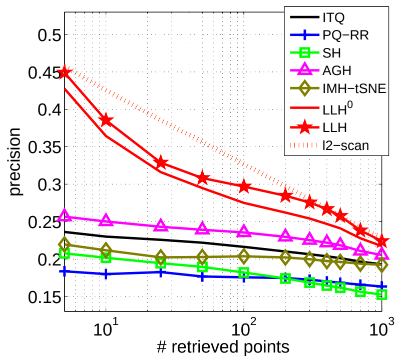

(d)

Figure 10. Comparison to Group-B on **MNIST**; (a) Averaged precision at top 500 retrieved points vs. the number of bits; the number of top retrieved images for (b) 16-bit, (c) 32-bit, and (d) 64-bit codes.


(a)

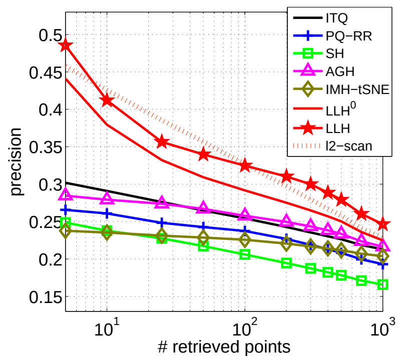
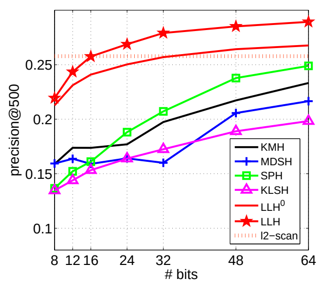
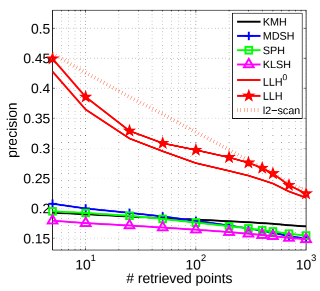
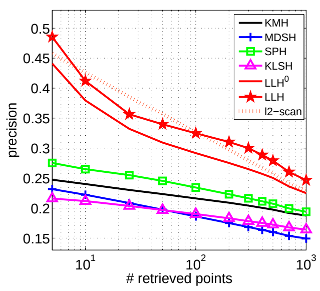

(b)

Figure 11. Comparison to Group-A on **CIFAR**; Averaged precision vs. the number of top retrieved images for (a) 16-bit and (b) 32-bit codes.


(a)


(b)


(c)

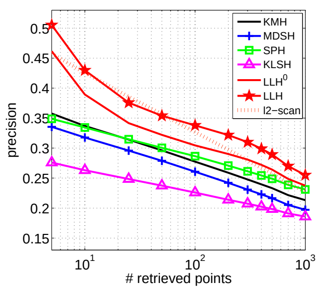
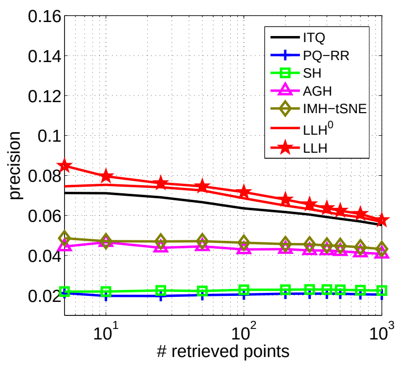
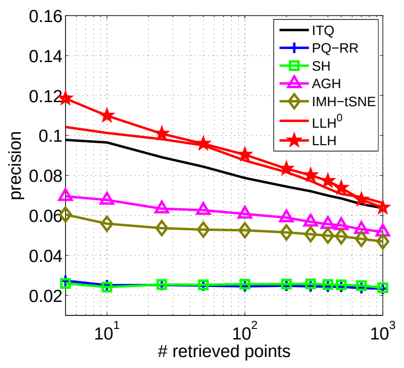

(d)

Figure 12. Comparison to Group-B on **CIFAR**; (a) Averaged precision at top 500 retrieved points vs. the number of bits; the number of top retrieved images for (b) 16-bit, (c) 32-bit, and (d) 64-bit codes.

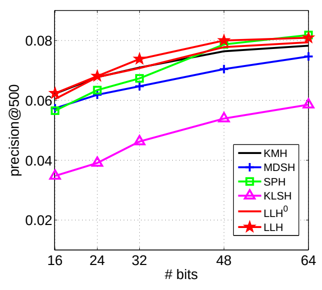


(a)

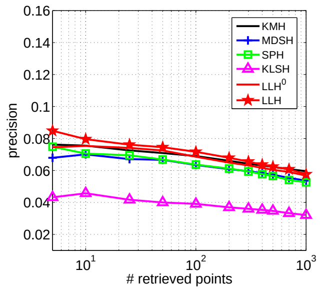


(b)

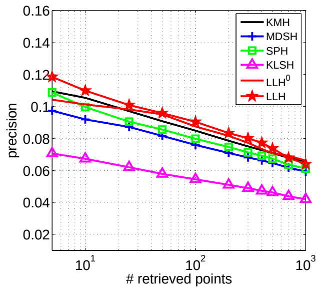
Figure 13. Comparison to Group-A on **ImageNet-200K**; Averaged precision vs. the number of top retrieved images for (a) 16-bit and (b) 32-bit codes.



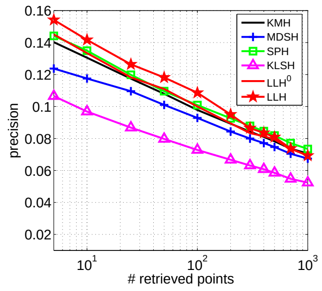
(a)



(b)



(c)



(d)

Figure 14. Comparison to Group-B on **ImageNet-200K**; (a) Averaged precision at top 500 retrieved points vs. the number of bits; the number of top retrieved images for (b) 16-bit, (c) 32-bit, and (d) 64-bit codes.

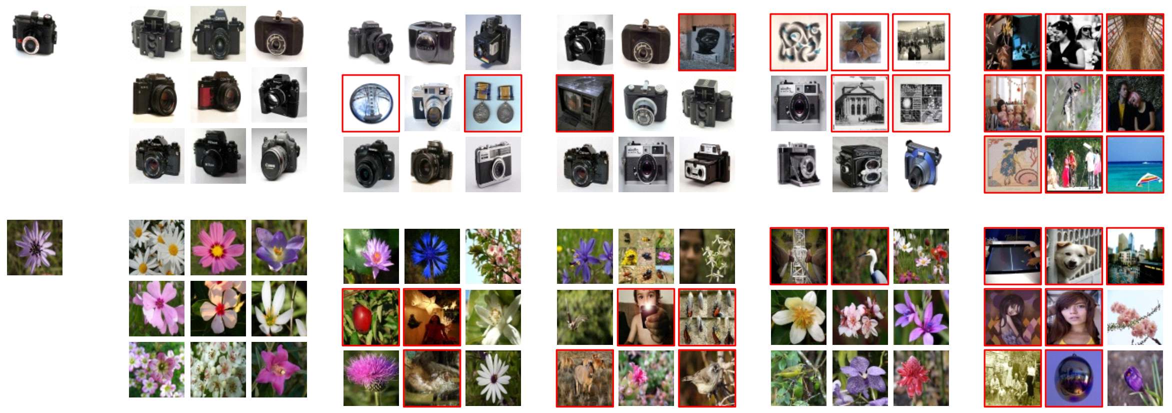
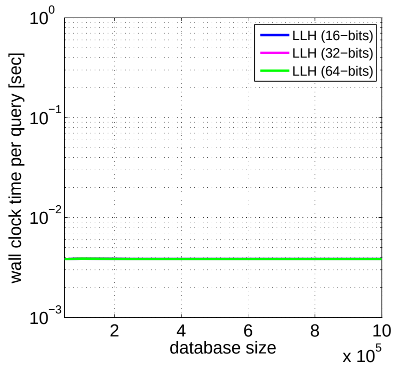
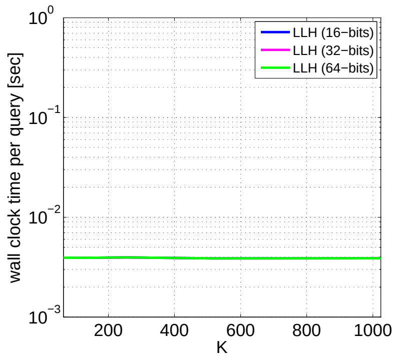

(a) query (b) LLH (c) KMH (d) MDSH (e) SPH (f) KLSH

Figure 15. Top retrieved images on **MIRFLICKR-1M** using 64-bit codes. Red border denotes false positive.

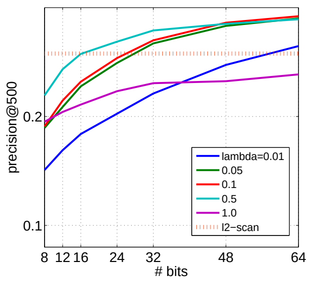


(a)

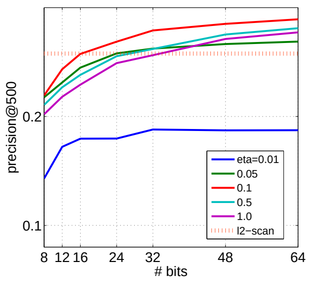


(b)

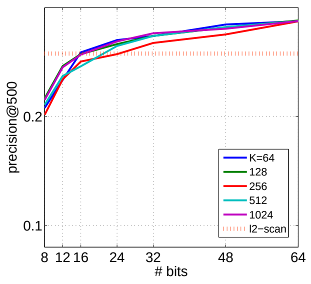
Figure 16. Scalability of the proposed out-of-sample extension algorithm on **MIRFLICKR-1M**; Averaged binary code generation time per query vs. (a) database size n and (b) K.



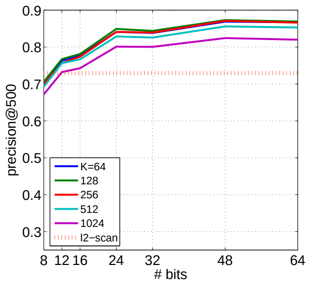
(a)



(b)



(c)



(d)

Figure 17. Parameter sensitivity; (a) λ, (b) η, and (c) K on **CIFAR** and (d) K on **MNIST**.