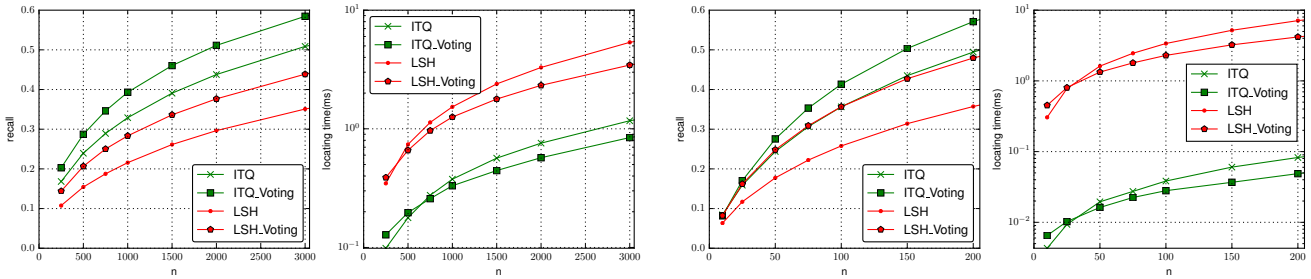

(a) LSH and ITQ on SIFT-1M

(b) LSH and ITQ on MNIST

Figure 2: Results on SIFT-1M and MNIST with different base hashing methods.

the detailed information of the datasets are summarized in Table 1. 32-bit hash codes and 10-NN graph are used for all the hashing methods. Constructing the graph costs 293s on SIFT-1M and 23s on MNIST respectively using 4 threads in the offline stage.

Table 1: Detailed information of the datasets

Dataset	Dimension	Base number	Query number
SIFT-1M	128	1,000,000	10,000
MNIST	784	60,000	10,000

4.2 Experimental Results

The key parameter is voting threshold (i.e., m). We set $m = 0$ to denote the boosted hashing method and the impact of m based on ITQ is shown in Table 2. We can see $m = 2$ can significantly improve the effectiveness, thus being used as our default setting. Although $m \geq 3$ can further improve the results, it will decrease efficiency at the same time. Note that when $m = 1$, our voting scheme reduces to one-iteration neighborhood expansion [5] . The results show that simple neighborhood expansion is not robust, as it may introduce noise and decrease the effectiveness on some dataset.

Table 2: Recall@100 based on ITQ with different m

m	0	1	2	3
SIFT-1M	0.101	0.090	0.119	0.129
MNIST	0.356	0.360	0.413	0.417

As we can see in Figure 2, on both datasets, our approach can boost corresponding hashing methods significantly and consistently in terms of different recall@n, which demonstrates the effectiveness of our approach. We also find that our approach usually reduces the locating time with the same number of candidates, leading to good search efficiency. For example, on SIFT-1M, our approach can improve the recall@1000 of ITQ from 0.329 to 0.393. Meanwhile, the locating time decreases from 0.376ms to 0.258ms. Note that under very small n, the location time might increase a little. In this case, the visiting reduction will be smaller than the additional voting cost in our approach, leading to a little bit higher locating time.

The memory overhead of our approach is small as compared with the size of original dataset (i.e., 488M for SIFT-1M and 179M for MNIST). For example, with the aggregated hash table based on ITQ, the additional overhead is 63M on SIFT-1M and 3.7M on MNIST in the case of $k = 10$.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we introduce k-NNs into hashing techniques and develop a novel search scheme namely neighborhood voting to enhance the search effectiveness of existing hashing methods. We also propose a novel data structure called aggregated hash table for efficient implementation. Experimental results show that our search scheme can significantly improve the search effectiveness while having good efficiency over different existing hashing techniques. In the future, we want to test our idea over other search schemes and investigate whether some theoretical guarantees can be provided.

6 ACKNOWLEDGMENTS

This work was funded by the 973 Program of China under Grant No. 2014CB340401, the National Natural Science Foundation of China (NSFC) under Grants No. 61425016, 61472401, 61722211, 61872338 and 20180290, the Youth Innovation Promotion Association CAS under Grants No. 20144310 and 2016102, and the National Key R&D Program of China under Grants No. 2016QY02D0405.

REFERENCES

[1] Wei Dong, Charikar Moses, and Kai Li. 2011. Efficient k-nearest neighbor graph construction for generic similarity measures. In *WWW*. ACM, 577–586.
[2] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. 2013. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *TPAMI* 35, 12 (2013), 2916–2929.
[3] Kiana Hajebi, Yasin Abbasi-Yadkori, Hossein Shahbazi, and Hong Zhang. 2011. Fast approximate nearest-neighbor search with k-nearest neighbor graph. In *IJCAI*, Vol. 22. 1312.
[4] Piotr Indyk and Rajeev Motwani. 1998. Approximate nearest neighbors: towards removing the curse of dimensionality. In *STOC*. ACM, 604–613.
[5] Zhongming Jin, Debing Zhang, Yao Hu, Shiding Lin, Deng Cai, and Xiaofei He. 2014. Fast and accurate hashing via iterative nearest neighbors expansion. *IEEE transactions on cybernetics* 44, 11 (2014), 2167–2177.
[6] Yu A Malkov and Dmitry A Yashunin. 2016. Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs. *arXiv preprint arXiv:1603.09320* (2016).
[7] Marius Muja and David G Lowe. 2012. Fast matching of binary features. In *CRV*. IEEE, 404–410.
[8] Mohammad Norouzi, Ali Punjani, and David J Fleet. 2012. Fast search in hamming space with multi-index hashing. In *CVPR*. IEEE, 3108–3115.
[9] Antonio Torralba, Rob Fergus, and Yair Weiss. 2008. Small codes and large image databases for recognition. In *CVPR*. IEEE, 1–8.
[10] Jingdong Wang, Heng Tao Shen, Jingkuan Song, and Jianqiu Ji. 2014. Hashing for similarity search: A survey. *arXiv preprint arXiv:1408.2927* (2014).