Supplementary document: Inductive Hashing on Manifolds[*]

Fumin Shen[‡†], Chunhua Shen[†], Qinfeng Shi[†], Anton van den Hengel[†], Zhenmin Tang[‡]
[†] The University of Adelaide, Australia [‡] Nanjing University of Science and Technology, China

In this document, we present the detailed analysis of the proposed prototype algorithm and the approximation approach in the main paper. We also shows the complete precision and recall curves for CIFAR-10 and MNIST.

1. Prototype algorithm and analysis

Assuming we have the embedding $Y := \{\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_n\}$ for the entire training data $X := \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$, one can predict the embedding for a new query point $\mathbf{x}_q$ via

$$\mathbf{y}_q = \frac{\sum_{i=1}^n \mathrm{w}(\mathbf{x}_q, \mathbf{x}_i)\mathbf{y}_i}{\sum_{i=1}^n \mathrm{w}(\mathbf{x}_q, \mathbf{x}_i)}. \tag{1}$$

However, n is often too massive (millions) to obtain embedding Y at the first place and computing $\mathrm{w}(\mathbf{x}_q, \mathbf{x}_i)$ for all data is too expensive. We will show that the following prototype algorithm is able to approximate $\mathbf{y}_q$ using only a small subset of Y well. This Prototype Algorithm is based on entropy numbers defined below,

Definition 1 (Entropy numbers[3]). *Given any $Y \subseteq \mathbb{R}^r$ and $m \in \mathbb{N}$, the m-th entropy number $\epsilon_m(Y)$ of Y is defined as*

$$\epsilon_m(Y) := \inf\{\epsilon > 0 | \mathcal{N}(\epsilon, Y, \|\cdot - \cdot\|) \leq p\},$$

where $\mathcal{N}$ is the covering number.

This means $\epsilon_m(Y)$ is the smallest radius that Y can be covered by less or equal to m balls.

1.1. Prototype Algorithm

Inspired by Theorem 27 of [3], we construct a Prototype Algorithm below. We use m balls to cover Y, thus obtain m disjoint nonempty subsets $Y_1, Y_2, \cdots, Y_m$ such that for any $\epsilon > \epsilon_m(Y)$, $\forall j \in \{1, \cdots, m\}, \exists \mathbf{c}_j \in \mathbb{R}^r$, s.t. $\forall \mathbf{y} \in Y_j, \|\mathbf{c}_j - \mathbf{y}\| \leq \epsilon$ and $\bigcup_{j=1}^m Y_j = Y$. We can see that each Y_j naturally forms a cluster with the center $\mathbf{c}_j$ and the index set $I_j = \{i | \mathbf{y}_i \in Y_j\}$.

Let $\alpha_i = \frac{\mathrm{w}(\mathbf{x}_q, \mathbf{x}_i)}{\sum_{j=1}^n \mathrm{w}(\mathbf{x}_q, \mathbf{x}_j)}$ and $C_j = \sum_{i \in I_j} \alpha_i$. For each cluster index set $I_j, j = 1, \cdots, m$, we randomly draw $\ell_j = \lfloor mC_j + 1 \rfloor$ many indices from I_j proportional to their weight α_i. That is, for $\mu \in \{1, \cdots, \ell_j\}$, the μ-th randomly drawn index $u_{j,\mu}$,

$$\Pr(u_{j,\mu} = i) = \frac{\alpha_i}{C_j}, \forall j \in \{1, \cdots, m\}.$$

We construct $\hat{\mathbf{y}}_q$ as

$$\hat{\mathbf{y}}_q = \sum_{j=1}^m \frac{C_j}{\ell_j} \sum_{\mu=1}^{\ell_j} \mathbf{y}_{u_{j,\mu}}. \tag{2}$$

Lemma 2. *There is at most $2m$ many unique $\mathbf{y}_{u_{j,\mu}}$ in $\hat{\mathbf{y}}_q$.*

[*]Appearing in IEEE Conf. Computer Vision and Pattern Recognition, 2013. F. Shen's contribution was made when he was visiting The University of Adelaide.

Proof. $\sum_{j=1}^{m} \ell_j \leq \sum_{j=1}^{m}(pC_j + 1) = 2m$. □

Lemma 3. *The following holds*

$$\mathrm{E}\left[\hat{\mathbf{y}}_q\right] = \mathbf{y}_q, \mathrm{Var}\left(\hat{\mathbf{y}}_q\right) \leq \frac{\epsilon^2}{m}. \tag{3}$$

Proof.

$$\begin{aligned}
\mathrm{E}\left[\hat{\mathbf{y}}_q\right] &= \mathrm{E}\left[\sum_{j=1}^{m} \frac{C_j}{\ell_j} \sum_{\mu=1}^{\ell_j} \mathbf{y}_{u_{j,\mu}}\right] = \sum_{j=1}^{m} \frac{C_j}{\ell_j} \sum_{\mu=1}^{\ell_j} \mathrm{E}\left[\mathbf{y}_{u_{j,\mu}}\right] \\
&= \sum_{j=1}^{m} \frac{C_j}{\ell_j} \sum_{\mu=1}^{\ell_j} \sum_{i \in I_j} \frac{\alpha_i}{C_j} \mathbf{y}_i = \sum_{j=1}^{m} \sum_{i \in I_j} \alpha_i \mathbf{y}_i = \mathbf{y}_q. \\
\mathrm{Var}\left(\hat{\mathbf{y}}_q\right) &= \sum_{j=1}^{m} \sum_{\mu=1}^{\ell_j} \mathrm{Var}\left(\frac{C_j}{\ell_j} \mathbf{y}_{u_{j,\mu}}\right) \leq \sum_{j=1}^{m} \frac{C_j^2}{\ell_j^2} \sum_{\mu=1}^{\ell_j} \epsilon^2 \\
&= \sum_{j=1}^{m} \frac{C_j^2}{\ell_j} \epsilon^2 \leq \sum_{j=1}^{m} \frac{C_j^2}{mC_j} \epsilon^2 = \frac{\sum_{j=1}^{m} C_j^2}{m} \epsilon^2 = \frac{\epsilon^2}{m}.
\end{aligned}$$

□

Above shows the mean is preserved and variance is small.

Theorem 4. *For any even number $n' \leq n$. If Prototype Algorithm uses n' many non-zero $\mathbf{y} \in Y$ to express $\hat{\mathbf{y}}_q$, then*

$$\Pr[\|\hat{\mathbf{y}}_q - \mathbf{y}_q\| \geq t] < \frac{2(\epsilon_{\frac{n'}{2}}(Y))^2}{n't^2}. \tag{4}$$

Proof. Via Chebyshev's inequality and Lemma 3, we know

$$\Pr\left(\|\hat{\mathbf{y}}_q - \mathbf{y}_q\| \geq k\sqrt{\mathrm{Var}\left(\hat{\mathbf{y}}_q\right)}\right) \leq \frac{1}{k^2}.$$

Let $t = k\sqrt{\mathrm{Var}\left(\hat{\mathbf{y}}_q\right)}$ and $\epsilon \to \epsilon_{\frac{n'}{2}}(Y)$ yields the theorem. □

Corollary 5. *For an even number n', any $\epsilon > \epsilon_{\frac{n'}{2}}(Y)$, any $\delta \in (0, 1)$ and any $t > 0$, if $n' \geq \frac{2\epsilon^2}{\delta t^2}$, then with probability at least $1 - \delta$,*

$$\|\hat{\mathbf{y}}_q - \mathbf{y}_q\| < t.$$

Proof. Via Theorem 4, we know that for $\epsilon > \epsilon_{\frac{n'}{2}}(Y)$, $\Pr[\|\hat{\mathbf{y}}_q - \mathbf{y}_q\| \geq t] < \frac{2\epsilon^2}{n't^2}$. Let $\delta \geq \frac{2\epsilon^2}{n't^2}$, we have $n' \geq \frac{2\epsilon^2}{\delta t^2}$. □

The quality of the approximation depends on $\epsilon_{\frac{n'}{2}}(Y)$ and n'. If data has strong clustering pattern, *i.e.* data within each cluster are very close to cluster center, we will have small $\epsilon_{\frac{n'}{2}}(Y)$, hence better approximation. Likewise, the bigger n' is, the better approximation is.

1.2. Approximation of Prototype Algorithm

For a query point $\mathbf{x}_q$, Prototype Algorithm samples from clusters and then construct $\hat{\mathbf{y}}_q$. The clusters can be obtained via clustering algorithm such as K-means. For each cluster, the higher $C_j = \sum_{i \in I_j} \alpha_i$, the more draws are made. At least one draw is made from each cluster. Since the n could be potentially massive, it is impractical to rank (or compute and keep a few top ones) α_i with in each cluster. Moreover, $\mathrm{w}(\mathbf{x}_q, \mathbf{x}_j)$ depends on $\mathbf{x}_q$ — for a different query point $\mathbf{x}_{q'}$, $\mathrm{w}(\mathbf{x}_{q'}, \mathbf{x}_i)$ may be very smaller even if $\mathrm{w}(\mathbf{x}_q, \mathbf{x}_i)$ is high. Thus we need to consider the entire X instead of a single $\mathbf{x}_q$.

Let $\alpha_i(\mathbf{x}_q) = \frac{\mathrm{w}(\mathbf{x}_q, \mathbf{x}_i)}{\sum_{j=1}^{n} \mathrm{w}(\mathbf{x}_q, \mathbf{x}_j)}$. Ideally, for each cluster, we want to select the $\mathbf{y}_i$ that has high overall weight $O_i = \sum_{\mathbf{x}_q \in X} \alpha_i(\mathbf{x}_q)$. For large scale X, the reality is that we don't have access to $\mathrm{w}(\mathbf{x}, \mathbf{x}')$ for all $\mathbf{x}, \mathbf{x}' \in X$. We only have

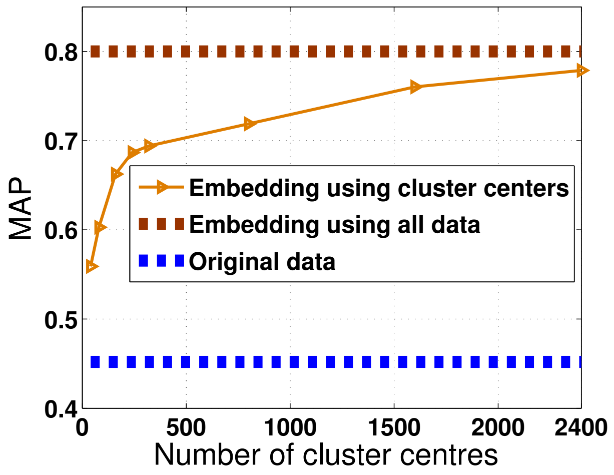

Figure 1: Euclidean search on 5,000 digital images using the original data, 32D t-SNE embeddings and approximate embeddings by IMH with K-means cluster centres.

limited information available such as cluster centers $\{\mathbf{c}_j, j \in \{1, \cdots, m\}\}$ and $\mathrm{w}(\mathbf{c}_j, \mathbf{x}), \mathbf{x} \in X$. Fortunately, the clustering result gives useful information about O_i. The cluster centers $\{\mathbf{c}_j, j \in \{1, \cdots, m\}\}$ have the largest overall weight w.r.t the points from their own cluster, *i.e.* $\sum_{i \in I_j} \mathrm{w}(\mathbf{c}_j, \mathbf{x}_i)$. This suggests we should select all cluster centers to express $\hat{\mathbf{y}}_q$. For an base set B, and any query point $\mathbf{x}_q$, we predict the embedding as

$$\hat{\mathbf{y}}_q = \frac{\sum_{\mathbf{x} \in B} \mathrm{w}(\mathbf{x}_q, \mathbf{x})\mathbf{y}}{\sum_{\mathbf{x} \in B} \mathrm{w}(\mathbf{x}_q, \mathbf{x})}. \tag{5}$$

To illustrate the ability of the method, we conduct searches on a 5,000 subset of MNIST by random sampling. It is clear that the proposed approximate method obtains much higher MAPs than the linear scan using the original features. With the increasing number of clusters, the proposed approximation approach gets closer performance to the search on the embedding obtained by using all data.

2. Experiments

We evaluate nine hashing algorithms including the proposed IMH-tSNE, IMH-LE and seven other state-of-the-art methods: PCAH [6], SH [7], AGH [5] and STH [8], BRE [4], ITQ [1], Spherical Hashing (SpH) [2]. We use the provided codes and suggested parameters according to the authors of these methods.

2.1. Results on CIFAR-10

Figure 2 shows the precision and recall curves of hamming ranking for the compared methods with different code lengths. From the top row, we see that ITQ, SpH and AGH obtain relatively high precisions when a small number of samples are returned, however their precisions drops significantly with the increasing number of retrieved samples. In contrast, IMH-tSNE achieve much higher precisions with relatively larger numbers of retrieved points. In terms of recall (shown in the bottom row of Figure 2), IMH-tSNE consistently outperform all other methods, and IMH-LE achieves good results with short bit lengths. STH achieves higher recall than AGH, which performs even worse than SH and PCAH with longer bits. Figure 3 details the precision-recall curves of these methods for different number of hash bits.

A few sample query images and the top retrieved samples are shown in Figure 4.

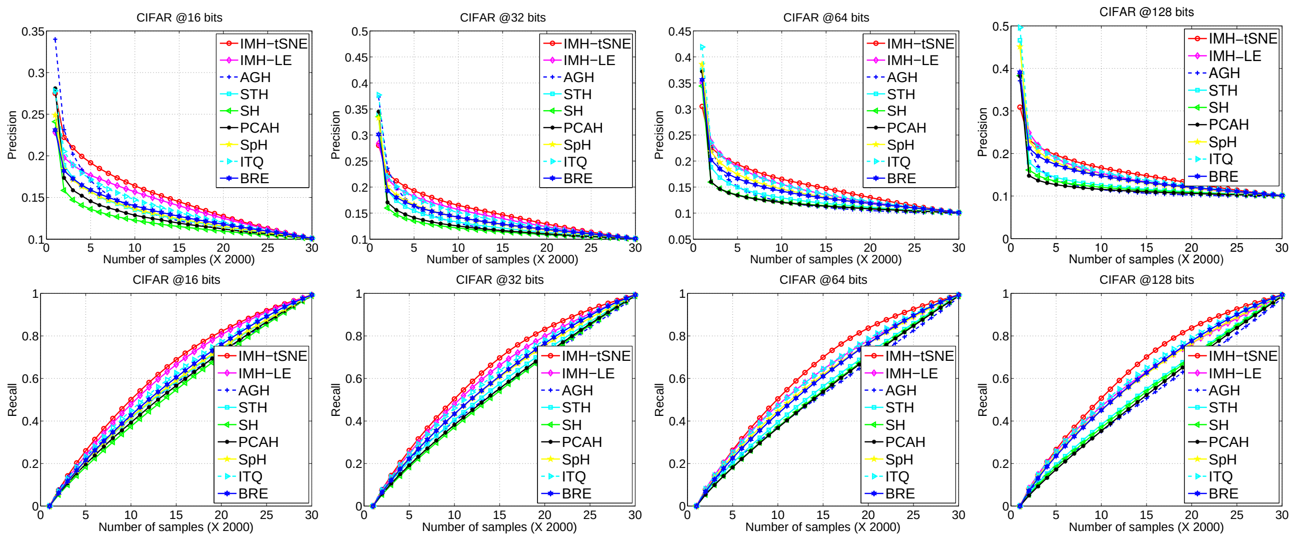

Figure 2: Comparison of different methods on CIFAR-10 based on precision (top row) and recall (bottom row) using different code lengths.

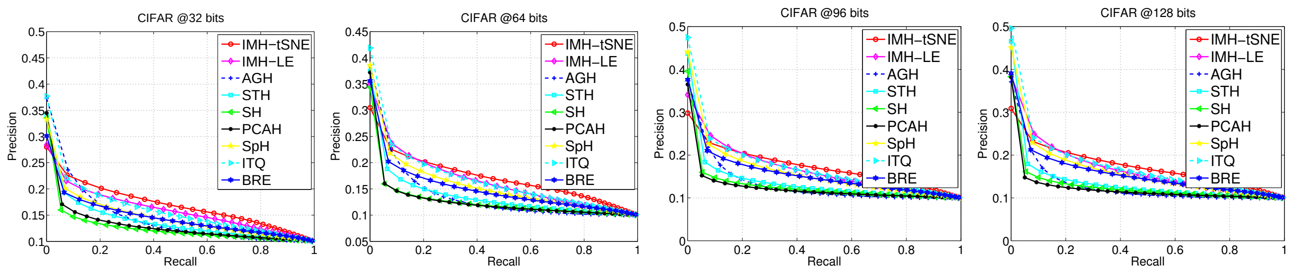

Figure 3: Comparison of different methods on CIFAR-10 based on precision-recall curves for different code lengths.

2.2. Results on MNIST

Figure 5 shows precision-recall curves for these methods for different numbers of hash bits. It is clearly seen that IMH-tSNE achieves much broader areas under the precision-recall curves. Specifically with 64 bits, given a 40% recall, IMH-tSNE obtains about 75% precision, while the best precision of all other methods obtained by STH is around 45%. Another interesting trend is that, unlike other competing methods, IMH-tSNE, ITQ and BRE consistently improve their performance as the number of bits increases (although the improvement becomes smaller with the increasing code lengths). This is consistent with the MAP results shown in the paper.

Figure 6 details the corresponding precision and recall curves. Consistent with Figure 5, IMH-tSNE demonstrates large margins over other methods for both precision and recall. ITQ, BRE and SpH obtain good results when with long bits, however are still inferior to IMH-tSNE. Among the LE based methods, IMH-LE perform close to AGH, both of which have higher precisions than STH and SH.

References

[1] Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2011.

[2] J. Heo, Y. Lee, J. He, S. Chang, and S. Yoon. Spherical hashing. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2012.

[3] R. Herbrich and R. C. Williamson. Algorithmic luckiness. *J. Mach. Learn. Res.*, 3:175–212, 2002.

[4] B. Kulis and T. Darrell. Learning to hash with binary reconstructive embeddings. In *Proc. Adv. Neural Inf. Process. Syst.*, 2009.

[5] W. Liu, J. Wang, S. Kumar, and S.-F. Chang. Hashing with graphs. In *Proc. Int. Conf. Mach. Learn.*, 2011.

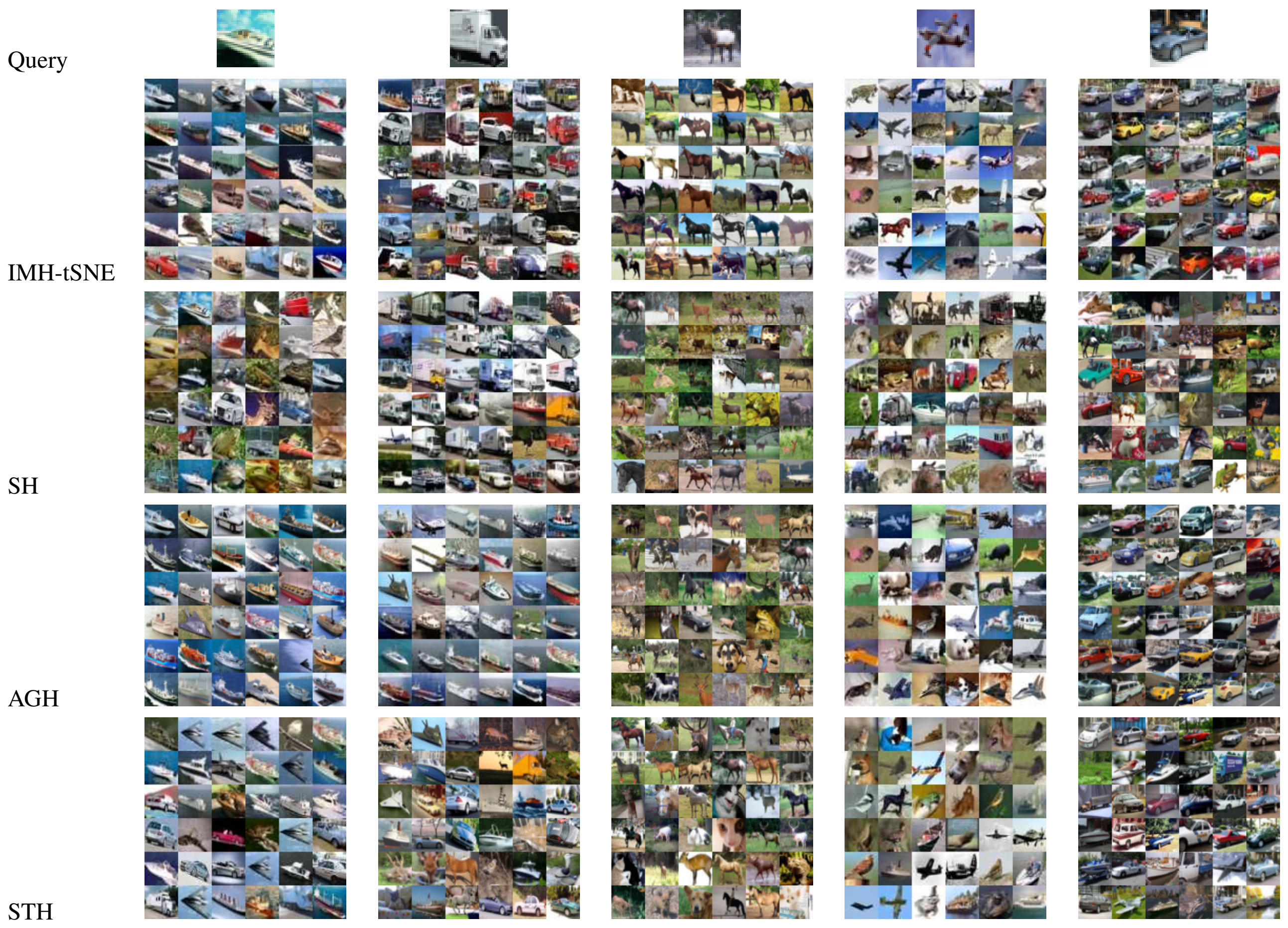

Figure 4: The query images (top row) and the retrieved images by IMH-tSNE, SH, AGH and STH with 32 hash bits.

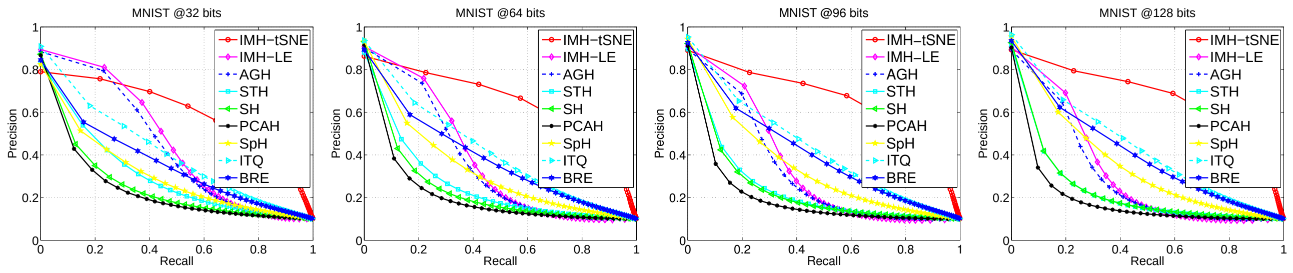

Figure 5: Comparison of different methods on MNIST based on precision-recall curves for different code lengths.

[6] J. Wang, S. Kumar, and S.-F. Chang. Semi-supervised hashing for large scale search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(12):2393 –2406, 2012.

[7] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *Proc. Adv. Neural Inf. Process. Syst.*, 2008.

[8] D. Zhang, J. Wang, D. Cai, and J. Lu. Self-taught hashing for fast similarity search. In *Proc. ACM SIGIR Conf.*, 2010.

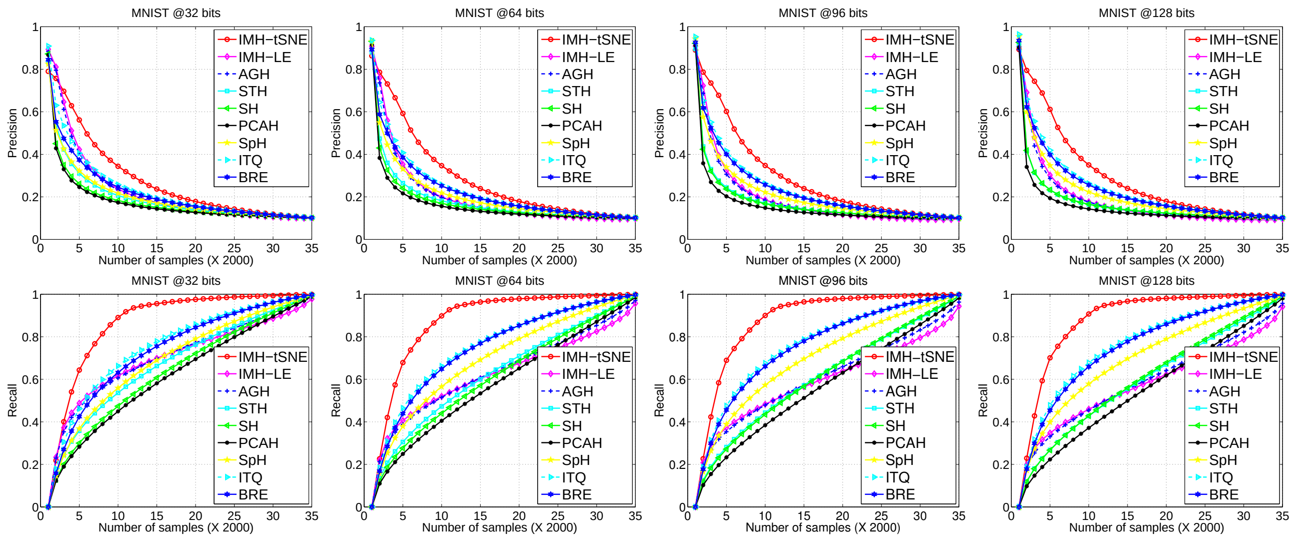

Figure 6: Comparison of different methods on MNIST based on precision and recall for different code lengths.