Supplementary Material

A. Distributional Derivative of Doubly Stochastic Neuron

Before we prove the lemma 3, we first introduce the chain rule of distributional derivative.

Lemma 6 *(Grubb, 2008) Let $u \in \mathcal{D}'(\Omega)$, we have*

1. **(Chain Rule I)** *The distribution derivative of $v = u \circ f$ for any $f(x) \in \mathcal{C}^1 : \Omega' \to \Omega$ is given by $Dv = Du\frac{\partial f}{\partial x}$.*

2. **(Chain Rule II)** *The distribution derivative of $v = f \circ u$ for any $f(x) \in \mathcal{C}^1(\mathbb{R})$ with f' bounded is given by $Dv = f'(u)Du$.*

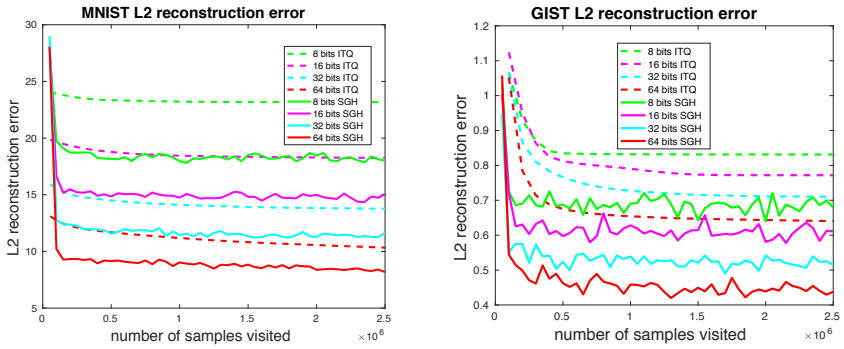
Proof of Lemma 3 Let $z = \sigma(W^\top x)$, we have

$$
\begin{aligned}
& D_z \mathbb{E}_{\epsilon,\xi}\left[\ell(f(z,\epsilon,\xi))\right] \\
\text{limit theorem (Grubb, 2008)} \quad &= \mathbb{E}_{\xi,\epsilon}\left[D_z \ell(f(z,\epsilon,\xi))\right] \\
\text{chain rule II} \quad &= \mathbb{E}_{\xi,\epsilon}\left[\nabla_f \ell(f(z,\epsilon,\xi)) D_z f(z,\epsilon,\xi)\right] \\
\text{chain rule I} \quad &= \mathbb{E}_{\xi,\epsilon}\left[\nabla_f \ell(f(z,\epsilon,\xi))\delta_\epsilon(z)\right] \\
\text{property of } \delta \text{ function} \quad &= \mathbb{E}_{\xi}\left[\nabla_f \ell(f(z,z,\xi))\right],
\end{aligned}
$$

Hence, $D_W \tilde{H}(\Theta, x) = \mathbb{E}_\xi\left[\nabla_f \ell(f(\sigma(W^\top x), \sigma(W^\top x), \xi))\right] \nabla\sigma(W^\top x) x^\top$. Plugging in the definition of f and using the fact that $\nabla\sigma(y) = \sigma(y) \bullet (1 - \sigma(y))$, leads to the desired result in (7). ■

B. More Experiments

B.1. Convergence of Distributional SGD and Reconstruction Error Comparison



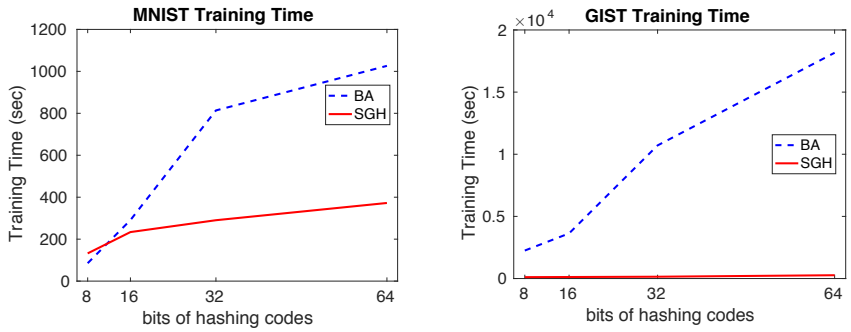
(a) MNIST (b) GIST−1M

Figure 4: L2 reconstruction error convergence on MNIST and GIST−1M of ITQ and SGH over the course of training with varying of the length of the bits (8, 16, 32, 64, respectively). The x-axis represents the number of examples seen by the training. For ITQ, it sees the training dataset once in one iteration.

We shows the reconstruction error comparison between ITQ and SGH on MNIST and GIST−1M in Figure 4. The results are similar to the performance on SIFT−1M. Because SGH optimizes a more expressive objective than ITQ (without orthogonality) and do not use alternating optimization, it find better solution with lower reconstruction error.

B.2. Training Time Comparison

We shows the training time comparison between BA and SGH on MNIST and GIST−1M in Figure 5. The results are similar to the performance on SIFT−1M. The proposed distributional SGD learns the model much faster.



(a) MNIST (b) GIST-1M

Figure 5: Training time comparison between BA and SGH on MNIST and GIST-1M.

B.3. More Evaluation on L2NNS Retrieval Tasks

We also use different RecallK@N to evaluate the performances of our algorithm and the competitors. We first evaluated the performance of the algorithms with Recall 1@N in Figure 6. This is an easier task comparing to $K = 10$. Under such measure, the proposed SGH still achieve the state-of-the-art performance.

In Figure 7, we set $K, N = 100$ and plot the recall by varying the length of the bits on MNIST, SIFT-1M, and GIST-1M. This is to show the effects of length of bits in different baselines. Similar to the Recall10@N, the proposed algorithm still consistently achieves the state-of-the-art performance under such evaluation measure.

C. Stochastic Generative Hashing For Maximum Inner Product Search

In Maximum Inner Product Search (MIPS) problem, we evaluate the similarity in terms of inner product which can avoid the scaling issue, *i.e.*, the length of the samples in reference dataset and the queries may vary. The proposed model can also be applied to the MIPS problem. In fact, the Gaussian reconstruction model also preserve the inner product neighborhoods. Denote the asymmetric inner product as $x^\top U h_y$, we claim

Proposition 7 *The Gaussian reconstruction error is a surrogate for asymmetric inner product preservation.*

Proof We evaluate the difference between inner product and the asymmetric inner product,

$$\|x^\top y - x^\top U^\top h_y\|_2 = \|x^\top \left(y - U^\top h_y\right)\|_2 \leqslant \|x\|_2 \|y - U^\top h_y\|_2,$$

which means minimizing the Gaussian reconstruction, *i.e.*, $-\log p(x|h)$, error will also lead to asymmetric inner product preservation. ■

We emphasize that our method is designed for hashing problems primarily. Although it can be used for MIPS problem, it is different from the product quantization and its variants whose distance are calculated based on lookup table. The proposed doubly stochastic neuron and the distributional derivative can be extended to quantization. This is out of the scope of this paper, and we will leave it as the future work.

C.1. MIPS Retrieval Comparison

To evaluate the performance of the proposed SGH on MIPS problem, we tested the algorithm on WORD2VEC dataset for MIPS task. Besides the hashing baselines, since KMH is the Hamming distance generalization of PQ, we replace the KMH with product quantization (Jegou et al., 2011). We trained the SGH with 71,291 samples and evaluated the performance with 10,000 query. Similarly, we vary the length of binary codes from 16, 32 to 64, and evaluate the performance by Recall 10@N. We calculated the ground-truth via retrieval through the original inner product. The performances are illustrated in Figure 8. The proposed algorithm outperforms the competitors significantly, demonstrating the proposed SGH is also applicable to MIPS task.

D. Generalization

We generalize the basic model to translation and scale invariant extension, semi-supervised extension, as well as coding with $h \in \{-1, 1\}^l$.

D.1. Translation and Scale Invariant Reduced-MRFs

As we known, the data may not zero-mean, and the scale of each sample in dataset can be totally different. To eliminate the translation and scale effects, we extend the basic model to translation and scale invariant reduced-MRFs by introducing parameter α to separate the translation effect and the latent variable z to model the scale effect in each sample x, therefore, the potential function becomes

$$E(x, h, z) = -\beta^\top h + \frac{1}{2\rho^2}(x - \alpha - U^\top(z \cdot h))^\top (x - \alpha - U^\top(z \cdot h)), \tag{10}$$

where $\cdot$ denotes element-wise product, $\alpha \in \mathbb{R}^d$ and $z \in \mathbb{R}^l$. Comparing to (2), we replace $U^\top h$ with $U^\top(z \cdot h) + \alpha$ so that the translation and scale effects in both dimension and sample are modeled explicitly.

We treat the α as parameters and z as latent variable. Assume the independence in posterior for computational efficiency, we approximate the posterior $p(z, h|x)$ with $q(h|x; W_h)q(z|x; W_z)$, where W_h, W_z denotes the parameters in the posterior approximation. With similar derivation, we obtain the learning objective as

$$\max_{U,\alpha,\beta,\rho;W_h,W_z} \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_{q(h|x_i)q(z|x_i)} \left[-E(x, h, z) - \log q(h|x_i) - \log q(z|x_i)\right]. \tag{11}$$

Obviously, the proposed distributional SGD is still applicable to this optimization.

D.2. Semi-supervised Extension

Although we only focus on learning the hash function in unsupervised setting, the proposed model can be easily extended to exploit the supervision information by introducing pairwise model, *e.g.*, (Zhang et al., 2014a; Zhu et al., 2016). Specifically, we are provided the (partial) supervision information for some pairs of data, *i.e.*, $\mathcal{S} = \{x_i, x_i, y_{ij}\}_{i,j}^M$, where

$$y_{ij} = \begin{cases} 1 & \text{if } x_i \in \mathcal{NN}(x_j) \text{ or } x_j \in \mathcal{NN}(x_i) \\ 0 & \text{o.w.} \end{cases},$$

and $\mathcal{NN}(x)$ stands for the set of nearest neighbors of x. Besides the original Gaussian reconstruction model in the basic model in (2), we introduce the pairwise model $p(y_{ij}|h_i, h_j) = \mathcal{B}(\sigma(h_i^\top h_j))$ into the framework, which results the joint distribution over x, y, h as

$$p(x_i, x_j, h_i, h_j, y_{ij}) = p(x_i|h_i)p(x_j|h_j)p(h_i)p(h_j)p(y_{ij}|h_i, h_j)^{\mathbf{1}_\mathcal{S}(ij)},$$

where $\mathbf{1}_\mathcal{S}(ij)$ is an indicator that outputs 1 when $(x_i, x_j) \in \mathcal{S}$, otherwise 0. Plug the extended model into the Helmholtz free energy, we have the learning objective as,

$$\max_{U,\beta,\rho;W} \frac{1}{N^2}\sum_{i,j=1}^{N^2} \Big(\mathbb{E}_{q(h_i|x_i)q(h_j|x_j)} \left[\log p(x_i, x_j, h_i, h_j)\right] + \mathbb{E}_{q(h_i|x_i)q(h_j|x_j)} \left[\mathbf{1}_\mathcal{S}(ij) \log p(y_{ij}|h_i, h_j)\right]$$
$$- \mathbb{E}_{q(h_i|x_i)q(h_j|x_i)} \left[\log q(h_j|x_j)q(h_j|x_i)\right]\Big),$$

Obviously, the proposed distributional SGD is still applicable to the semi-supervised extension.

D.3. $\{\pm 1\}$-Binary Coding

In the main text, we mainly focus on coding with $\{0, 1\}$. In fact, the proposed model is applicable to coding with $\{-1, 1\}$ with minor modification. Moreover, the proposed distributional SGD is still applicable. We only discuss the basic model here, the model can also be extended to scale-invariant and semi-supervised variants.

If we set $h \in \{-1, 1\}^l$, the potential function of basic reduced-MRFs (2) does not have any change, *i.e.*,

$$E(x, h) = -\beta^\top h + \frac{1}{2\rho^2}\left(x^\top x + h^\top U^\top U h - 2x^\top U h\right). \tag{12}$$

We need to modify the parametrization of $q(h|x)$ as

$$q(h|x) = \prod_{i=1}^{l} \sigma(w_i^\top x)^{\frac{1+h_i}{2}} \left(1 - \sigma(w_i^\top x)\right)^{\frac{1-h_i}{2}} . \tag{13}$$

Therefore, the doubly stochastic neuron becomes

$$f(z, \epsilon, \xi) := \begin{cases} 1 & \text{if } \sigma(z) > \epsilon \\ 2 \times \mathbf{1}_{\sigma(z)>\xi} - 1 & \text{if } \sigma(z) = \epsilon \\ -1 & \text{if } \sigma(z) < \epsilon \end{cases} .$$

With similar derivation, we have the distributional derivative of the objective w.r.t. W as

$$\nabla_W L_{sn} = 2\mathbb{E}_\xi \left[\nabla_f \ell(f(z, \sigma(z), \xi)) \nabla_z \sigma(z) x^\top\right] . \tag{14}$$

Plug these modification into the model and algorithm, we can learn a $\{-1, 1\}$-encoding function.

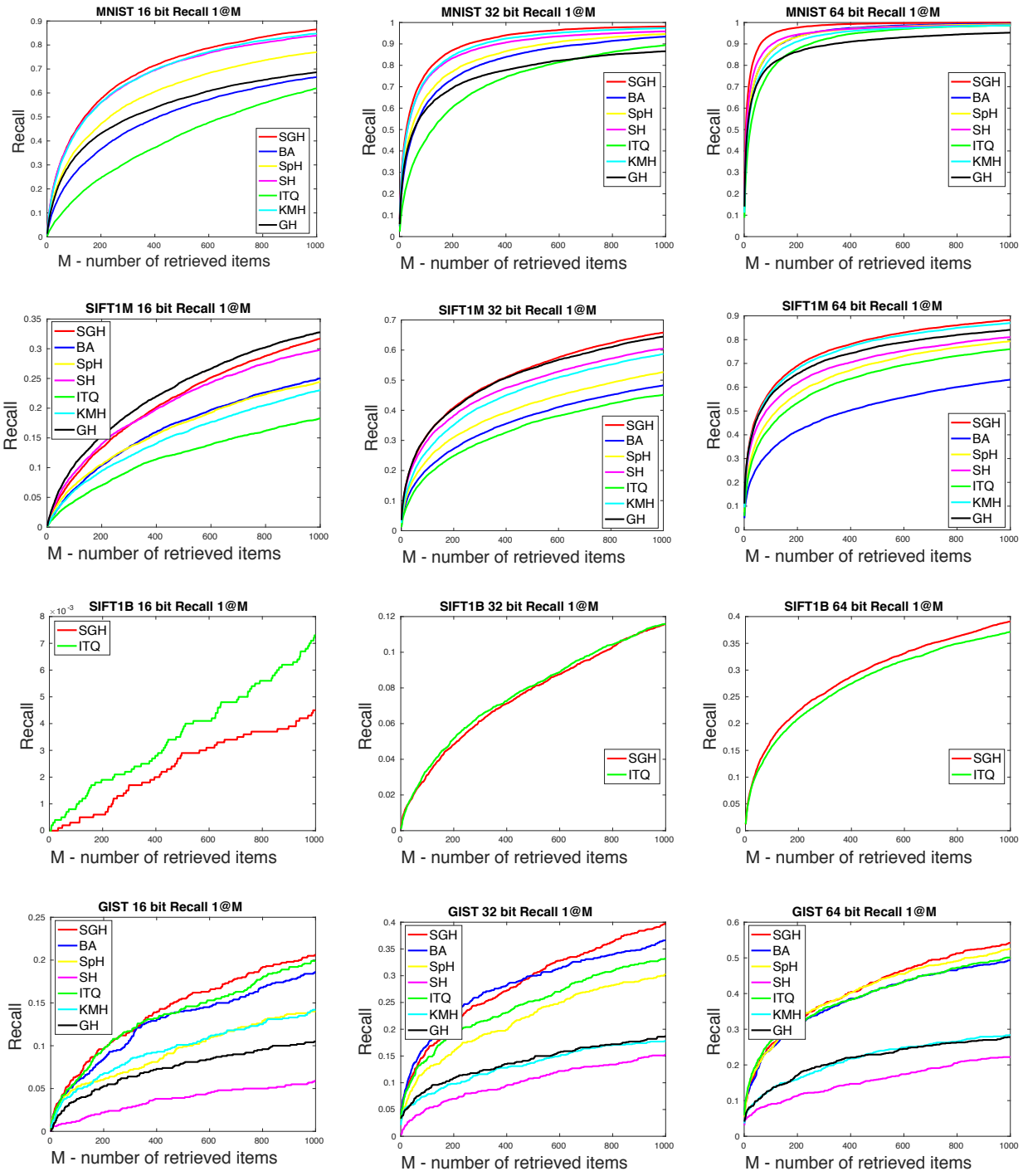
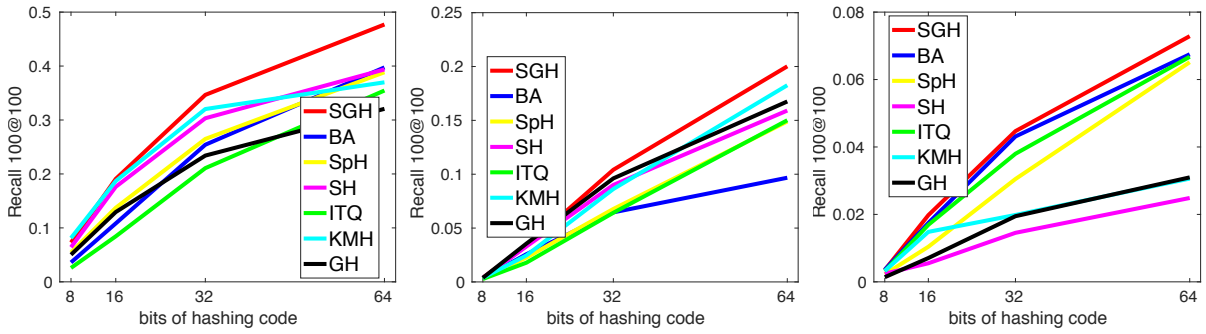


Figure 6: L2NNS comparison on `MNIST`, `SIFT-1M`, `SIFT-1B`, and `GIST-1M` with the length of binary bits from 16 to 64. We evaluate the performance with Recall $1@M$, where M increasing to 1000.



(a) L2NNS on MNIST (b) L2NNS on SIFT−1M (c) L2NNS on GIST−1M

Figure 7: L2NNS comparison on MNIST, SIFT−1M, and GIST−1M with Recall 100@100 for the length of bits from 8 to 64.

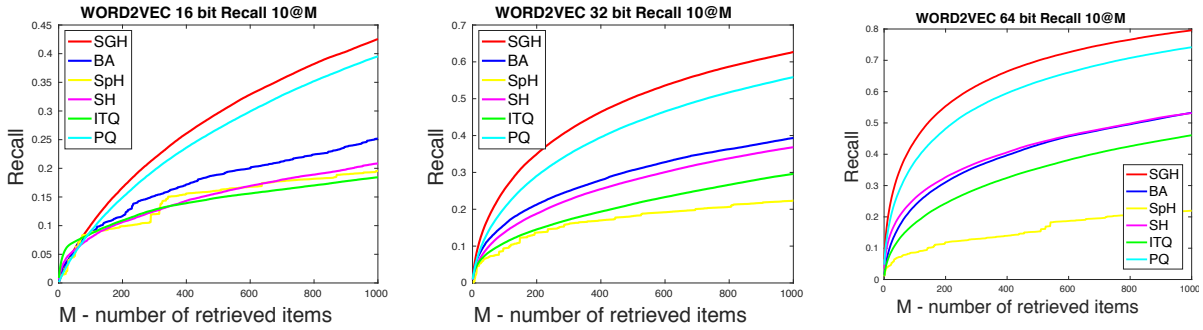


Figure 8: MIPS comparison on WORD2VEC with the length of binary bits from 16 to 64. We evaluate the performance with Recall 10@M, where M increasing to 1000.