

Efficient Cross-Modal Retrieval via Deep Binary Hashing and Quantization (Supplemental material)

Yang Shi
yang.shi@rakuten.com

Rakuten Inc.
San Mateo, USA

Young-joo Chung
youngjoo.chung@rakuten.com

In this supplemental material, we present preliminaries of hashing and quantization in Section 1. In Section 2, we show approximation error analysis, and computation and memory analysis of HQ. Finally, we provide data statistics, implementation details and additional experiments about retrieval efficiency in Section 3.

1 Preliminaries

Binary hashing Given input $x \in \mathbb{R}^n$, we compute binary code $h_x = H(x) \in \mathbb{R}^n$, where $H(\bullet)$ is a function that maps continuous value to $\{+1, -1\}$. The similarity measurement is Hamming distance:

$$\text{dist}(h_x, h_y) = \text{sum}(h_x \text{ XOR } h_y) \quad (1)$$

The larger the value, the greater the dissimilarity between the two.

Quantization Given input $x \in \mathbb{R}^n$, we compute the quantization $x \approx Cb_x$, where $C \in \mathbb{R}^{n \times k}$ is the dictionary book, $b_x \in \mathbb{R}^k$ is the index indicator for x . We assume the input is assigned to only one entry of the dictionary: $\|b_x\|_0 = 1$. Different inputs will share the same C but will have different index indicators. We use Asymmetric Quantizer Distance (AQD) to measure quantization similarities:

$$AQD(x, y) = x^T(Cb_y) \quad (2)$$

A greater value correlates with a greater similarity between the two. Generally, quantization preserves more information than binary codes, though it is slower at searching step since AQD requires more computation than Hamming distance.

© 2021. The copyright of this document resides with its authors.
It may be distributed unchanged freely in print or electronic forms.

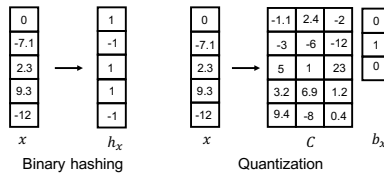


Figure 1: Binary hashing and quantization.

Model	Memory	Computation
Lossless	$O(32Nn)$	$O(Nn)$
Quantization	$O(32mkn + Nm \log_2 k)$	$O(mkn + Nm)$
Binary hash	$O(Nn)$	$O(Nn)$
HQ	$O(Nn + 32mkn + Nm \log_2 k)$	$O(Nn + mkn + \alpha Nm)$

Table 1: Memory and computation analysis for different models. We assume the total number of items in database is N , dense feature dimension n , each of the m quantization dictionary book has length k

2 Analysis

In this section, we theoretically show that minimizing HQ’s loss function can provide the best compact code error. We also analyze computation and memory complexity of HQ and other hashing methods in the cross-modal retrieval task.

2.1 Approximation Error Analysis

We use h_i, h_j, Cb_i, Cb_j and f_i, f_j to represent binary codes, quantization codes and dense features for data i and data j from different modalities.

Hashing error Even though Hamming distance is used for computing binary code distance, we use the relaxed Euclidean distance to compute the error bound since it is convex.

$$\begin{aligned}
& |d(h_i, h_j) - d(f_i, f_j)| \\
&= |||h_i - h_j||_F^2 - ||f_i - f_j||_F^2| \\
&= |||sgn(f_i) - sgn(f_j)||_F^2 - ||f_i - f_j||_F^2|
\end{aligned} \tag{3}$$

Minimizing Equation 3 is equal to minimizing the distance between $sgn(f_i)$ and f_i . In other words, the continuous feature should be close to +1 or -1. We achieved this by adding a *tanh* layer after the final feature layer.

Quantization error Since AQD approximates inner product distance, we compare AQD with direct inner product results using continuous features f .

$$|AQD(i, j) - d(f_i, f_j)| = |(f_i)^T Cb_j - (f_i)^T f_j| \leq \|f_i\| \|f_j - Cb_j\| \tag{4}$$

Because $\|f_i\|$ is a fixed scalar which does not effect the relative distance measurement, we only need to minimize $\|f_j - Cb_j\|$: the distance between dense feature and quantization feature. This term already exists in the loss function L_q . Thus, we are guaranteed to get best quantization error by minimizing our proposed loss function.

2.2 Computation and Memory Analysis

In this section, we compare the computation and memory efficiency of HQ with the method which only uses quantization.

Memory requirement for quantization is $O(32mkn + Nm \log_2 k)$: since we have m dictionary books and each book has k n -length features, we need $O(32mkn)$ to store the dictionary books when data is in single-precision float(32bits). For each data point, we will use $O(m \log_2 k)$ to store the m one-of- k indicator ($\log_2 k$ bits) b . We assume we have N data points.

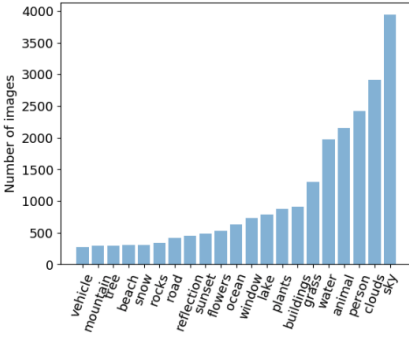


Figure 2: NUS-WIDE Image-label distributions.

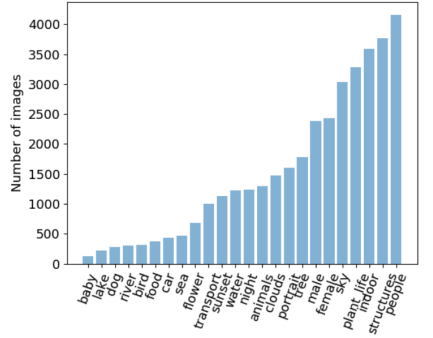


Figure 3: MIR-Flickr Image-label distributions.

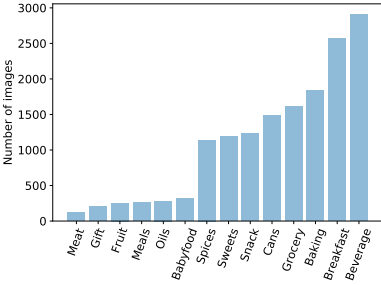


Figure 4: Amazon Image-label distributions.



Figure 5: Retrieval example in MIR-Flickr.

Memory requirement for binary hashing feature is $O(Nn)$. Note that if we use a lossless model which uses continuous features, it will be $O(32Nn)$.

For the quantization model, we can pre-compute the dot product between the query and each dictionary with $O(mnk)$ operations and store it in a temporary look-up table, then we need $O(m)$ additions to compute the AQD distance between query and each data point with the look-up table. Overall, we need $O(mnk + Nm)$ operations to complete one query retrieval with the quantization model, while we need $O(Nn)$ for the hashing model. We show the complexity comparison between the lossless model, the quantization model, and the hashing model in Table 1. Quantization and hashing are more memory efficient than the lossless model.

3 Experiments

3.1 Dataset

We used NUS-WIDE dataset [10], MIR-Flickr dataset [9] and Amazon Review [8] for experiments. The NUS-WIDE public web image dataset contains images associated with textual tags. Besides, each image-tag pair is annotated with one or multiple labels from 81 concept labels, such as “nature,” “water,” and “sunset.” MIR-Flickr consists of 25,000 images collected from the Flickr website. Each image-tag sample belongs to at least one of the 24 labels,

Models	#Params.	Time	
		$I \rightarrow T$	$T \rightarrow I$
DJSRH	120M	3.19	2.69
JDSH	62M	2.73	3.03
HQ	62M	2.62	2.36

Table 2: The retrieval time (seconds) on NUSWIDE (a total of 2,100 queries over 13,650 data points, Number of parameters is computed using model.parameters() in Pytorch. All models use 128-dim hash code).

such as “sky,” “car,” “lake.” We selected Amazon Grocery and Gourmet Food category and used their image-title pairs as training/test data. Each image-title belongs to at least one of the 14 categories, such as “meat,” “fruit,” and “spices”.

Figure 5 shows an image-to-text retrieval example. Here, we retrieved a group of text tags in the database for a given query image. From the retrieved tags, we list up their associated labels. Since the highlighted labels exist in the image’s labels (i.e. female, people, plant-life, tree), we consider the retrieval is correct.

3.2 Implementation Details

To extract image features in NUS-WIDE and MIR-Flickr, we initialized image feature extraction networks with pre-trained VGGNet-19 [14] for Section 5.3 and with AlexNet [15] pre-trained on the ImageNet dataset [16] for Section 5.4. Amazon image features are provided by the original paper [17]. In all datasets, final image features are 128-length vectors. For text features, we followed previous work [18] and extracted 1,000, 1,386 and 1,000 most frequently used tags/words for NUS-WIDE, MIR-Flickr and Amazon, respectively, and created bag-of-word vectors for text inputs. These vectors were converted to the final 128-length text features using a two-layer MLP.

We cross-validated the hyper parameters and finally set $\lambda_q = 0.0001$, $\lambda_{sim} = 70$, $\lambda_h = 0.01$, $\lambda_b = 0.01$ for NUS-WIDE experiments, $\lambda_q = 0.0001$, $\lambda_{sim} = 50$, $\lambda_h = 0.01$, $\lambda_b = 0.01$ for MIR-Flickr experiments and $\lambda_q = 0.001$, $\lambda_{sim} = 10$, $\lambda_h = 0.01$, $\lambda_b = 0.001$ for Amazon experiments. λ_q was set especially small because we observed quantization loss decreasing much faster than other losses. $\alpha N = 100$ is set for all experiments. All experiments were performed on a NVIDIA GeForce GTX 1080 Ti. All codes were written with PyTorch 1.1.0.

3.3 Additional experiments about retrieval efficiency

Table 2 shows the number of model parameters and retrieval time of DJSRH, JDSH, and HQ on NUS-WIDE dataset. Note that JDSH requires less parameters than DJSRH, for it is a simplified DJSRH. While HQ and JDSH have the similar number of parameters, HQ’s retrieval time is faster. This proves HQ’s efficiency.

References

- [1] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, and Yantao Zheng Zhiping Luo. A real-world web image database from national university of singapore. *CIVR*, 2009.

- [2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.
- [3] Mark J. Huiskes and Michael S. Lew. The mir flickr retrieval evaluation. In *MIR '08: Proceedings of the 2008 ACM International Conference on Multimedia Information Retrieval*, New York, NY, USA, 2008. ACM.
- [4] Qing-Yuan Jiang and Wu-Jun Li. Deep cross-modal hashing. *CVPR*, pages 3232–3240, 2017.
- [5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Neurips*, 2012.
- [6] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, page 43–52, New York, NY, USA, 2015. Association for Computing Machinery.
- [7] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.