Central Similarity Quantization for Efficient Image and Video Retrieval — Supplementary Material

1. Proof on Hash Center Validity from H_{2K}

When $K < m \leqslant 2K$ in Algorithm 1, we use the combination of two Hadamard matrices $H_{2K} = [H_K, -H_K]^\top$ to construct the hash centers. Here, we prove that the rows of H_{2K} can also be valid hash centers in the K-dimensional Hamming space. According to Definition 1, we know if the Hamming distance between any two row vectors of H_{2K} is equal to or lager than $K/2$, the row vectors of H_{2K} are valid hash centers.

We consider following three cases for Hamming distance between any two row vectors h_i and h_j in H_{2K}:

1. Both of the two row vectors h_i and h_j belong to the upper half or below half of $H_{2K} = [H_K, -H_K]^\top$, *i.e.*, $h_i, h_j \in H_K$ or $h_i, h_j \in -H_K$. So h_i and h_j are still orthgonal with each other with an inner product of $\langle h_i, h_j \rangle = 0$. We get the Hamming distance $D_H(h_i, h_j) = \frac{1}{2}(K - \langle h_i, h_j \rangle) = \frac{K}{2}$;

2. One of the two row vectors belongs to H_K, and the other one belongs to $-H_k$. We assume $h_i \in H_K$ and $h_j \in -H_K$. If $h_i \neq -h_j$, the two row vectors are still orthgonal with each other, thus $D_H(h_i, h_j) = \frac{K}{2}$.

3. One of the two row vectors belongs to H_K, and the other one belongs to $-H_k$, but $h_i = -h_j$. Thus the inner product is $\langle h_i, h_j \rangle = -K$, and $D_H(h_i, h_j) = K$.

We summarize these three situations as following:

$$D_H(h_i, h_j) \begin{cases} = \frac{K}{2} & \text{if } h_i, h_j \in H_K \text{ or } -H_K \\ = \frac{K}{2} & \text{if } h_i \in H_K, h_j \in -H_K, h_i \neq -h_j \\ = K & \text{if } h_i \in H_K, h_j \in -H_K, h_i = -h_j \end{cases} \tag{S.1}$$

So the average Hamming distance is larger than $K/2$, and the row vectors in H_{2K} are valid hash centers in K-dimensional Hamming space.

For multi-label data, the K/2 distance lower bound still holds if they do not share any semantic label, which can be proven by doing a sampling simulation on the Hadamard matrix. Assuming that we randomly sample six hash centers a_1, a_2, a_3 and b_1, b_2, b_3, the centroid of a_1, a_2, a_3 is c_a, and c_b is the centroid of b_1, b_2, b_3. The c_a includes two parts: the first part c_{a1} comes from the common/major bits of a_1, a_2, a_3, while the second part c_{a2} is sampled from Bernoulli distribution, as well as c_b. If we directly let length(c_{a1})=length(c_{b1}), length(c_{a2})=length(c_{b2}), then we can get $\mathbb{E}[D_H(c_{a1}, c_{b1}) + D_H(c_{a2}, c_{b2})] = K/2$.

The Hadamard method is superior to the Bernoulli method because the mutual distance between hash centers can be maximized when sampling from Hadamard matrix (the proof is in supplementary materials), while Bernoulli distribution cannot guarantee to generate hash centers with maximized mutual distance. So, we adopt the Hadamard method when $m <= 2K \& K = 2^n$ and the Bernoulli method when $m > 2K$.

2. Jointly Learning with Pairwise Similarity

Given the semantic hash centers $\mathcal{C}' = \{c'_1, c'_2, \ldots, c'_N\}$ and pairwise similarity label $\mathcal{S} = \{s_{ij}\}$, we can formulate central similarity and pairwise similarity based learning together to optimize the deep hashing functions. Recall the similarity label $s_{ij} = 1$ indicates the data pairs x_i and x_j are similar. The Maximum Likelihood (ML) estimation of hash codes $\mathcal{H} = [h_1, ..., h_N]$ for all training data $\mathcal{X}$ with label $\mathcal{S}$ can be obtained by maximizing the following likelihood probability:

$$P(\mathcal{C}', \mathcal{S}|\mathcal{H}) = P(\mathcal{S}|\mathcal{H}) P(\mathcal{C}'|\mathcal{S}, \mathcal{H}). \tag{S.2}$$

Since we build the hash centers based on $\mathcal{S}$, the $P(\mathcal{C}'|\mathcal{S})$ is known and can be treated as constant. Equation (S.2) thus becomes $P(\mathcal{C}', \mathcal{S}|\mathcal{H}) \propto P(\mathcal{S}|\mathcal{H}) P(\mathcal{C}'|\mathcal{H})$. Then the log likelihood can be written as

$$\begin{aligned} &\log P(\mathcal{C}', \mathcal{S}|\mathcal{H}) \propto \log P(\mathcal{C}'|\mathcal{H}) + \log P(\mathcal{S}|\mathcal{H}) \\ &= \sum_{c'_i \in \mathcal{C}'} \log P(c'_i|h_i) + \sum_{s_{ij} \in \mathcal{S}} \log P(s_{ij}|h_i, h_j), \end{aligned} \tag{S.3}$$

where the first RHS term represent the central similarity and the second RHS term is the pairwise similarity. The central similarity loss L_C has been given in Sec. 3.3. For the pairwise similarity term in Equation (S.3), we use the inner product of the hash codes to measure the probability of the similarity labels.

Recall the Hamming distance and inner product for any two hash codes h_i and h_j satisfies: $D_H(h_i, h_j) = \frac{1}{2}(K - \langle 2h_i - \mathbf{1},\ 2h_j - \mathbf{1}\rangle)$, where $\mathbf{1}$ is the all-one vector. We use inner product to replace the Hamming distance and define $P(s_{i,j}|h_i, h_j)$, the conditional probability of $s_{i,j}$, as follows:

$$P(s_{ij}|h_i, h_j) = \begin{cases} \sigma(\langle 2h_i - \mathbf{1},\ 2h_j - \mathbf{1}\rangle), & s_{ij} = 1, \\ 1 - \sigma(\langle 2h_i - \mathbf{1},\ 2h_j - \mathbf{1}\rangle), & s_{ij} = 0, \end{cases} \tag{S.4}$$

or equivalently,

$$\begin{aligned} &P(s_{ij}|h_i, h_j) \\ &= \sigma(\langle 2h_i - \mathbf{1}, 2h_j - \mathbf{1}\rangle)^{s_{ij}} (1 - \sigma(\langle 2h_i - \mathbf{1}, 2h_j - \mathbf{1}\rangle))^{1 - s_{ij}}, \end{aligned} \tag{S.5}$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the Sigmoid function. This logistic regression-alike formulation satisfies that the smaller the Hamming distance $D_H(h_i, h_j)$, the larger the inner product $\langle 2h_i - 1, 2h_j - 1)\rangle$ and the larger the conditional probability $P(1|h_i, h_j)$. This means that the pairs h_i and h_j have a large probability to be classified as similar. Otherwise, the pairs would be classified to be dissimilar ($P(0|h_i, h_j)$ is large). After algebraic calculations, maximizing the above likelihood can be equivalently written as minimizing the following the pairwise similarity loss L_P is computed as:

$$\begin{aligned} L_P = \sum_{s_{ij} \in S} &(\log(1 + \exp(\langle 2h_i - \mathbf{1}, 2h_j - \mathbf{1}\rangle)) \\ &- s_{ij} \langle 2h_i - \mathbf{1}, 2h_j - \mathbf{1}\rangle). \end{aligned} \tag{S.6}$$

Putting all the pieces together, we obtain the following jointly optimization

$$\min_{\Theta} L_T = L_C + \lambda_1 L_Q + \lambda_2 L_P. \tag{S.7}$$

where L_Q is the quantization loss, which has been given in the main text. Because the original method will only generate continuous centers, the quantization loss is used to make the learned centers be binarized for the purpose of hashing. In the experiment section of the main text, we also present and discuss performance of jointly learning by combining both L_C and L_P in the first ablation study. The λ_1 and λ_2 are weights for the three losses, which are obtained by grid search.

3. Implementation Details

3.1. Framework of Central Similarity Quantization

The framework of Central Similarity Quantization is shown in Fig. S.1. The input of CSQ is $\{(x_i, x_j, c_i, c_j)\}$. Here c_i and c_j are the hash centers for x_i and x_j respectively. CSQ takes this input and outputs compact hash codes through the following deep hashing pipeline: 1) a 2D or 3D CNN sub-network to extract the data representation for image or video data, 2) a hash layer with three fully-connected layers and activation functions to project high dimensional data features to hash codes in the Hamming space, 3) a central similarity loss L_C for central similarity-preserving learning, where all hash centers are defined in the Hamming space, making hash codes converge on corresponding centers. and 4) a quantization loss L_Q for improving binarization. For image hashing, we adopt 2D CNN to learn image features. For video hashing, we adopt 3D CNN to learn video features.

3.2. Implementation details for image retrieval

We implement CSQ by adopting AlexNet [4] and ResNet [2] architecture as 2D CNN for image feature learning. For fair comparison, the four baseline deep methods also use the same network with the same configurations. We fine-tune the four convolution layers *conv1* to *conv4* with learning rate 1e-5, which inherits from AlexNet or ResNet model pre-trained on the ImageNet. We never touch the test data in pre-training. We train the hash layer from scratch with 20 times learning rate than the convolution layers. We use the Adam solver [3] with a batch size of 64 and the hyper-parameters $\lambda_1 = 0.05$ and $\lambda_2 = 0.2$ are obtained by grid searching.

3.3. Implementation details for video retrieval

We employ **MFN** [1] as 3D CNN for video feature learning. The CSQ is first pre-trained on action classification task to learn video features, and we copy the parameters of 3D convolution layers. Then we fine tune the convlutional layers with learning rate 5e-4, and train the hash layer with 5 times learning rate than the 3D convolution layers. We use mini-batch stochastic gradient decent (SGD) with 0.9 momentum. The batch size is 32 and weight decay parameters is 1e-4. We train on two TITAN X GPU (12G) and takes around 16 hours for UCF101 and 9 hours for HMDB51.

4. Hash Center Learning

We give the loss functions to learn hash cener by three different methods:

Face Center We adopt the method to learn centers as the common face recognition [6]. The learned centers are $\{c_1, c_2, ..., c_p\}$, where p is number of category. Then the center c_i can be updated in each mini-batch according to the following function:

$$\Delta c_i = \frac{\sum_{j=1}^{m} \delta(l_j = i)(c_i - x_j))}{1 + \sum_{j=1}^{m} \delta(l_j = i)}, \tag{S.8}$$

where l_j is the semantic label of data x_j. So the center loss of "Face Center" is: $L_{FC} = \Delta c_i$. We learn the hash center

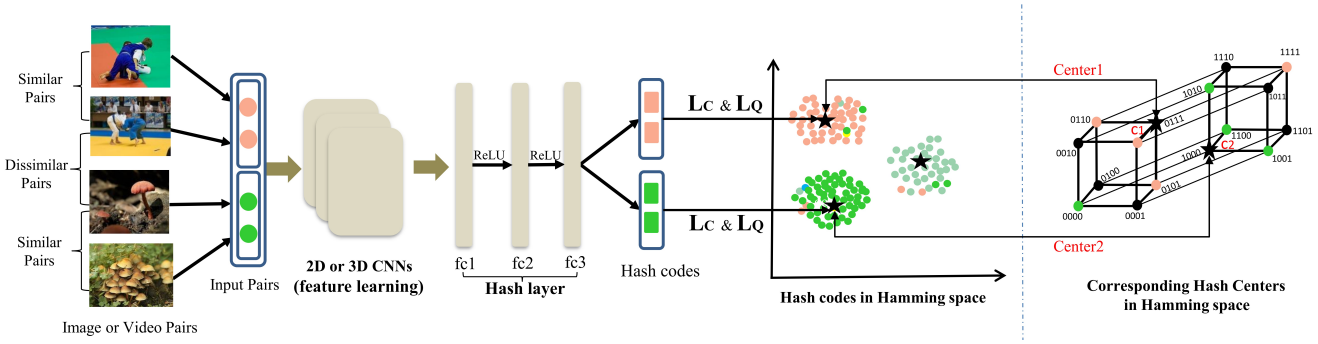

Figure S.1. Framework of proposed Central Similarity Quantization (CSQ). CSQ takes as input similar and dissimilar pairs (images or videos). For image or video data, we use different types of CNNs for feature learning. After passing through a hash layer, similar pairs converge to a common center and dissimilar pairs converge to different centers by adding central similarity constraint. The convergent targets are the hash centers (Center1 and Center2) in the Hamming space.

and hash codes by following function:

$$\min_{\Theta} L_T = L_C + \lambda_1 L_Q + \lambda_2 L_{FC}, \tag{S.9}$$

where L_Q is quantization loss, and L_C is central similarity loss, as defined in our paper. Hyper-parameters $\lambda_1 = 0.02$ and $\lambda_2 = 1.5$ are obtained by grid search.

Magnet Center Magnet center are learned by K-means [5]. The centers $\{c_1, c_2, ..., c_p\}$ are updated in each mini-batch according to the K-means algorithm as:

$$c_i = \arg\min_{c_i} \sum_{k=1}^{K} \|c_i - \mu_k^c\|^2, \tag{S.10}$$

where K is the number of similar data points in each mini-batch and u_k is the representation of data x_k.

So once obtain centers in each mini-batch, the total loss function in Magnet Center hashing is:

$$\min_{\Theta} L_T = L_C + \lambda_1 L_Q, \tag{S.11}$$

Hyper-parameters $\lambda_1 = 0.005$ is obtained by grid search.

RepMet Center The loss function to learn centers $\{c_1, c_2, ..., c_p\}$ is

$$L_{RC} = |\min D_H(u_k, c_i) - \min_{j \neq i} D_H(u_k, c_j) + \alpha|_+, \tag{S.12}$$

where u_k is the feature of x_k, and the center of x_k is c_i. $|.|_+$ is the ReLU function. This loss function try to ensure at least α margin between the Hamming distance of data point to its center and to other closest centers. We set $\alpha = K/2$ to keep a large distance between data points.

Similarity with Face Center and Magnet Center, the total loss function in RepMet Center hashing is:

$$\min_{\Theta} L_T = L_C + \lambda_1 L_Q + \lambda_2 L_{RC}, \tag{S.13}$$

Hyper-parameters $\lambda_1 = 0.001$ and $\lambda_2 = 0.9$ are obtained by grid search.

Robustness experiment of hash centers When applying our method as Algorithm 1 to generate hash centers, we can sample different rows of the Hadamard matrix to generate hash centers. To show CSQ performs consistently well for different hash center choices, we evaluate its performance for five different combinations of hash centers. From the results in Table 1, we can validate the robustness of CSQ to hash center choices.

Table 1. Run five times for different hash center choices. The results are mean ± std for mAP@64bits.

Dataset	ImageNet	MS COCO	HMDB51	UCF101
mAP	0.868±0.13	0.860±0.27	0.579±0.35	0.873±0.13

5. Visualization of Hash Centers

We visualize some generated hash centers from algorithm 1 in this section. The 64×64 Hadamard matrix H_{64} is shown as Fig. S.2. The hash centers of 64-bit for the five datasets we used are constructed by H_{64} and $H_{2K} = [H_{64}, -H_{64}]^\top$ as Algorithm 1.

For NUS_WIDE, we only sample 21 most frequent categories for experiments. Because of $16 < 21 < 32$, all the hash centers with bits of 16, 32 and 64 for NUS_WIDE is constructed by Hadamard matrix H_{16}, H_{32} and H_{64}. For the other four datasets, the 64-bit hash centers are constructed by H_{64} and H_{2K}, but 16-bit and 32-bit hash centers are constructed by sampling from Bernoulli distributions. We give the illustration of the hash centers of 16-bit, 32-bit and 64-bit for ImageNet in Fig. S.4 and Fig. S.5. The hash centers for other three datasets are similar. In Fig. S.4 and Fig. S.5, every row means one hash center for one category in ImageNet.

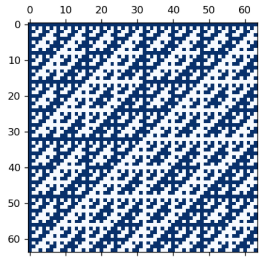

Figure S.2. Hadamard matrix H_{64}: 64×64.

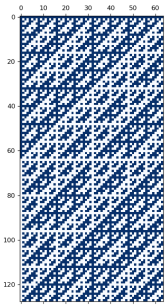

Figure S.3. Hadamard matrix $H_{2K} = [H_{64}, -H_{64}]^T$: 128×64.

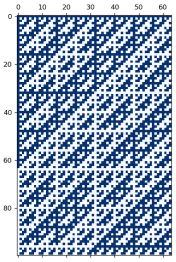

Figure S.4. Hash centers of ImageNet@64bit: 100×64. Each row represents one hash center, and there are totally 100 hash centers. The number of column represents the dimension of the hash center.

(a) 32bit (b) 16bit

Figure S.5. Hash centers of ImageNet@32bit and @16bit. The size of (a) is 100×32; the size of (b) is 100×16. Each row represents one hash center, and the number of column represents the dimension of the hash center.

References

[1] Y. Chen, Y. Kalantidis, J. Li, S. Yan, and J. Feng. Multi-fiber networks for video recognition. *arXiv preprint arXiv:1807.11195*, 2018. 2

[2] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2

[3] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 2

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 2

[5] O. Rippel, M. Paluri, P. Dollar, and L. Bourdev. Metric learning with adaptive density discrimination. *arXiv preprint arXiv:1511.05939*, 2015. 3

[6] Y. Wen, K. Zhang, Z. Li, and Y. Qiao. A discriminative feature learning approach for deep face recognition. In *European Conference on Computer Vision*, pages 499–515. Springer, 2016. 2