DHWP: LEARNING HIGH-QUALITY SHORT HASH CODES VIA WEIGHT PRUNING

Zeyu Ma[1], Yuhang Guo[2], Xiao Luo[2], Chong Chen[4], Minghua Deng[2], Wei Cheng[3], Guangming Lu[1*]*

[1]*School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, China*
[2]*School of Mathematical Sciences, Peking University, Beijing, China*
[3]*University of Chinese Academy of Sciences, Beijing, China* [4]*Alibaba Group, Hangzhou, China*

ABSTRACT

Hashing is widely used in large-scale image retrieval because of its efficiency in storage and computation. Although longer hash codes can lead to higher search accuracy, the retrieval cost increases linearly with the increase of the number of hash bits. Most deep hashing methods suffer from the problem of trivial solutions and usually result in highly correlated redundant hash bits in practice, which limits the performance. To obtain short hash codes with high quality for fast and accurate image retrieval, we propose a novel framework named Deep Hashing via Weight Pruning (DHWP). DHWP first trains the model with relatively long hash codes. Then it obtains shorter codes gradually by weight pruning based on four different criteria. The framework of DHWP can be applied to most deep supervised hashing models, which helps remove those redundant hash bits while retaining the representation ability of long hash codes. Extensive experimental results on two widely used benchmark datasets show that DHWP outperforms the existing state-of-the-art methods, especially for short hash codes.

***Index Terms*—** Deep hashing, Image retrieval, Supervised hashing

1. INTRODUCTION

Hashing, which transforms high-dimensional features into binary codes, is one of the most widely used approximate nearest neighbor search methods for its efficiency of computation and storage [1]. For the powerful representation capabilities of deep learning, more and more deep hashing methods were proposed, which can help achieve end-to-end hashing code learning [2, 3, 4, 5].

Short hash codes can improve the efficiency of calculation and storage, but also bring a lot of information loss. Therefore, searching directly with short hash codes is difficult to guarantee the recall rate. A coarse-to-fine search strategy [6] is usually used for rapid but accurate image retrieval. Although longer hash codes will lead to higher search accuracy, the retrieval cost increases linearly with the number of hash bits. And most deep hashing methods suffer from the trivial solution problem and usually lead to highly correlated redundant hash bits in practice, which could limit the performance. To address this problem, [7] adopts the neuron merging layer to reduce the redundancy through merging neurons to balance the importance of each bit. However, the neuron fusion method is relatively complicated and it only considers supervised situations.

Neural network pruning aims to remove the useless weights or network structures while maintaining the original accuracy as much as possible, which has attracted much attention recently [8, 9, 10]. Neural network pruning has the superiority of reducing the redundancy and increasing the interpretability of the model while maintaining the performance of neural networks, which can be applied in many fields including bioinformatics and computer vision [11, 12].

Inspired by neural network pruning [11, 13], we propose a novel framework named Deep Hashing with Weight Pruning (DHWP) to obtain short hash codes with high quality. First, we train the model with a relatively larger number of hash bits. Then it keeps masking some hash bits based on one of four different criteria and fine-tunes the model progressively. Several considerations underlie the design of weight pruning. First, it has been proved that starting with training a large and over-parameterized network could provide a model with high performance by its stronger representation and optimization power. Second, the bit redundancy has a significant impact on the retrieval performance by experiments, which may produce noise at the end of training. Third, the code length is also a significant impact factor for the sake of speed and storage efficiency. Our contributions are summarized as follows:

- We propose a novel framework named DHWP, which introduces a novel mask layer to remove the redundant hash bits through weight pruning. To the best of our knowledge, DHWP is the first deep hashing method to take advantage of weight pruning.
- We also propose a progressive optimization strategy to

This work was supported in part by the NSFC fund 62176077, in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2019B1515120055, in part by the Shenzhen Key Technical Project under Grant 2020N046, in part by the Shenzhen Fundamental Research Fund under Grant JCYJ20210324132210025. Zeyu Ma and Yuhang Guo contribute equally to this paper. *Corresponding authors.

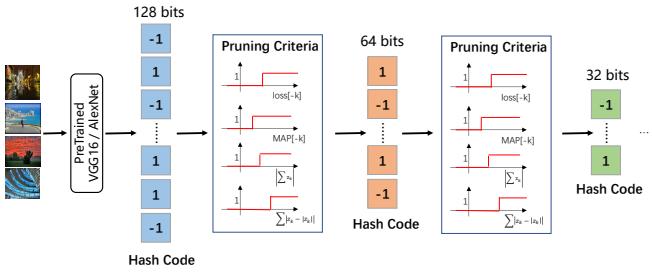


Fig. 1. Architecture of DHWP. DHWP first trains a given state-of-the-art model with relatively longer hash code, then gradually prunes the redundant hash bits by four different criterion to obtain high quality short hash codes.

gradually reduce the redundancy, generating high-quality hash codes with different lengths by a compressed deep hashing model.

- DHWP is a general framework that can be applied to most deep supervised hashing methods, which helps to get short hash codes with higher quality. Extensive experiments on two benchmark datasets show that DHWP improves the search performance of state-of-the-art methods, especially for short hash codes.

2. APPROACH

2.1. Problem Formulation

In our task, we are given a dataset with n points $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^{n}$, in which $\boldsymbol{x}_i$ is the feature of the i-th image with dimension $\mathcal{R}^d$. The similarity label $s_{ij} = 1$ implies the i-th image and the j-th image are similar, or otherwise $s_{ij} = 0$. For the supervised task, the similar pairs are constructed from the image labels, i.e., two images will be considered to be similar if they have at least one common label. Deep supervised hashing aims to learn a non-linear hash function: $f : \boldsymbol{x} \rightarrow \boldsymbol{h} \in \{-1, 1\}^K$, which encodes each sample $\boldsymbol{x}$ into compact $K-$ bit hash code $\boldsymbol{h}$ and we hope that the original similarity relationship $\{(\boldsymbol{x}_i, \boldsymbol{x}_j, s_{ij})\}$ between image pairs are well preserved. For computational consideration, the distance between different hash codes is Hamming distance, which is formulated as $\text{dis}_H(\boldsymbol{h}_i, \boldsymbol{h}_j) = \frac{1}{2}(K - \langle \boldsymbol{h}_i, \boldsymbol{h}_j \rangle)$,where $<,>$ denotes the inner product of hash codes.

2.2. Neural Network Architecture

Figure 1 demonstrates the architecture of the proposed Deep Hashing with Weight Pruning. We extend from the well-known image classification deep neural network architecture (e.g., AlexNet[14]) where the pre-trained weights on large-scale dataset ImageNet are available. We remove the last fully connected layer (classifier layer) in the chosen deep neural network model as the encoder of our model. The decoder is a new fully-connected layer with H hidden units, which outputs the H-dimensional continuous code $\boldsymbol{z}$ $(H > K)$. Afterward, we add a mask layer to prune the unit according to the given criteria introduced below.

$$\boldsymbol{e} = Encoder(\boldsymbol{x}) \tag{1}$$

$$\boldsymbol{z} = Decoder(\boldsymbol{e}) \tag{2}$$

$$u_k = z_k * m_k, k = 1, \cdots, H \qquad (Masklayer) \tag{3}$$

$$\boldsymbol{h} = Remove\_zero(sign(\boldsymbol{u})) \tag{4}$$

where $\boldsymbol{e}$ is the deep feature obtained from the encoder. Denote that the mask layer aims to mask the units that have little impact on the image retrieval performance at the end of the training. As a result, m_k is set to be 0 or 1, and $\sum_k m_k = K$. $m_k = 0$ means that the k-th units z_k are discarded and vice visa. Lastly, the K-dimension hash code $\boldsymbol{h}$ can be obtained by sign thresholding and removing the zeros of the output $\boldsymbol{u}$ in the mask layer.

2.3. Pruning Criterion

The pruning criteria are critical for the performance of our model. Here we propose four pruning criteria for DHWP. Considering that minimizing the loss function is our direct desire, we prune the units which have little impact on the loss function if removed. The loss-based criterion is formulated as

$$m_k = \begin{cases} 1 & loss[-k] > threshold \\ 0 & otherwise \end{cases}, \tag{5}$$

in which $loss[-k]$ is the loss without the k-th unit (i.e., the loss when $m_k = 0$). The threshold is set to $(100 - p)$-th percentile of all the $loss[-k]$ if we want to retain p percent of units. In the second MAP-based criterion, we utilize the mean average precision(MAP) instead of the loss function for pruning the units. The retrieval and database datasets are both training sets, and the units with little impact on the MAP are pruned

$$m_k = \begin{cases} 1 & MAP[-k] < threshold \\ 0 & otherwise \end{cases}, \tag{6}$$

in which $MAP[-k]$ is the MAP without the k-th unit. Inspired with Hash Boosting [15], we consider the bit balance (i.e., each bit has a 50% chance of being 1 or -1.) because the loss function and MAP have been considered in the training process. The bits with severe imbalance are pruned

$$m_k = \begin{cases} 1 & |\sum_{\boldsymbol{i}} z_{k,i}| < threshold \\ 0 & otherwise \end{cases}, \tag{7}$$

in which $z_{k,i}$ is the z_k for the i-th image $\boldsymbol{x}_i$ in the training set. If the bit meet the condition of bit balance, $|\sum_{\boldsymbol{i}} z_{k,i}|$ will approximate 0. Last, we also consider the quantization loss over the training set because it's expected that the output

of the decoder approximates 1 or -1. The units with small quantization loss are retained:

$$m_k = \begin{cases} 1 & \sum_{\boldsymbol{i}} |z_{k,i} - sign(z_{k,i})| < threshold \\ 0 & otherwise \end{cases} . \quad (8)$$

2.4. Progressive Optimization Strategy

Our model takes the iterative pruning strategy on the units of the decoder and drops the learning rate of each pruning step gradually for fine-tuning. First, the decoder outputs $H = K * d_1 * \cdots * d_m$ units, which means the whole model is large and over-parameterized. For each time, the $1/d_i$ of units are retained by pruning criteria. In other words, the number of valuing being 1 in the mask layer is $1/d_i$ of that before each pruning step and finally, only K units are left after m-th step. Since weight pruning is likely to lead to decreased performance, we then fine-tune the pruned model to regain the lost performance. The above two steps (i.e., pruning and fine-tuning) are iterated for m times and the final model is generated. DHWP first initializes with the weights of the pre-trained model in the encoder and adjusts the weights gradually by iterative pruning and fine-tuning. As a result, the learning rate in the encoder for fine-tuning is much lower than that in the decoder. In this way, the drawback of easily stopping at the local optima of the model with limited units could be overcome by our pruning strategy [16].

2.5. Implementation of the DHWP

Here we follow the DPSH [2] to minimize the pairwise similarity-preserving loss inferred from the log-likelihood. Specifically, we have supervised similarity signals $\{s_{ij}\}_{i,j=1}^{n}$. Taking the negative log-likelihood of the observed pairwise labels in similarity signals, the loss function of DPSH [2] is formulated as

$$\begin{aligned} \min \quad & \mathcal{L}_{\text{DPSH}} = \sum_{i=1}^{n}\sum_{j=1}^{n} -(s_{ij}\Omega_{ij} - log(1+e^{\Omega_{ij}})) \\ & +\eta \sum_{i=1}^{n} ||\boldsymbol{z}_i - sign(\boldsymbol{z}_i)|| \\ \text{s.t.} \quad & \boldsymbol{z}_i, \boldsymbol{z}_j \in \mathcal{R}^K \end{aligned} \quad (9)$$

in which $\Omega = \frac{1}{2}\boldsymbol{z}_i^T \boldsymbol{z}_j$ and η is a hyper-parameter. The second term is the quantization loss to push the output to 1 and -1. Note that the $\boldsymbol{z}_i$ is the output of the mask layer for the i-th image. If the j-th element of $\boldsymbol{z}_i$ is masked, the formulation does not need to be changed. As a result, the above formulation is our basic pairwise hashing loss. All d_i are between 1 and 2 because the proposed pruning strategy is a greedy method, and thus the pruned number should be smaller than half each time. Specifically, in our settings, the original code length is set to 64 and then the code length is pruned to be 48, 32, 24, 12 orderly. To make a comparison, we match our model with baseline, which is the basic model utilizing identical K units trained directly without pruning. The model is trained through the standard backpropagation algorithm [17]. Although DHWP is implemented based on the model following DPSH, the DHWP can also be integrated into other deep supervised methods. From the discussion above, it is evident that the framework of DHWP relies on the loss function, training map, bit balance, and quantization loss, which exists in most deep supervised methods. In general, the DHWP is a generalized and effective framework for deep hashing.

3. EXPERIMENTS

3.1. Datasets and Setup

CIFAR-10 [18] consists of 60000 images from 10 different categories. We randomly select 100 images as training images and 500 as queries for each class. All images except for the query set are used as the retrieval set. **NUS-WIDE** [19] contains 269,648 images and each of the images is annotated with multiple labels referring to 81 concepts. The subset containing the 21 most popular concepts is utilized for our task. We randomly select 2100 images as a test set; the remaining images are used as a retrieval set, and 10,500 images are randomly selected from the retrieval set as the training set.

The baseline is our proposed model with the same architecture of DHWP but without the mask layer (i.e., DHWP with corresponding bits outputs). Besides baseline, our method is compared with a list of state-of-the-art hashing methods including traditional methods and deep hashing methods. Traditional hashing methods include FastH [20], SDH [21], KSH [22]. Deep hashing methods including DQN [23], NINH [24], CNNH [25], DHN [26] and CIBhash [27].

The retrieval quality is evaluated by Mean Average Precision (MAP), Precision-Recall curves, and Top N precision curves. For NUS-WIDE, the MAPs are calculated on the top 5,000 returned images following [2]. Precision-recall curves can reveal the precision at different recall levels, which is a good indicator of overall performance. Top N precision curve is the precision curve in regard to the top N retrieved instances, which visualizes the performance in a different way.

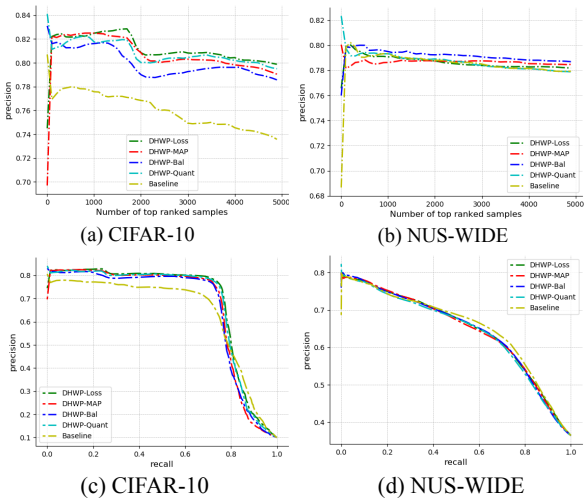
We optimize our model by mini-batch RMSprop with weight decay 10^{-5}. The mini-batch size is 24. Training images are all resized to 224×224 as the inputs. The initial code length is set to 64. η is set to 0.1 following [2].

3.2. Results

From table 1, we have the following observations: Compared with traditional methods, deep learning-based methods can always have better MAP performances, which indicates the strong ability of deep learning in image representation. Four weight pruning methods all achieve better performances compared with the baseline DPSH, and the improvement of MAP increases with the decrease of hash code length. For single-label dataset CIFAR-10, compared to the baseline, the MAP is about 3% to 5% higher when the length of the hash code is

Table 1. MAP results on datasets CIFAR-10 (@50,000) and NUS-WIDE (@5,000). DHWP-Loss, DHWP-MAP, DHWP-Bal and DHWP-Quant refer to weight pruning by training loss, MAP, bit balance and quantization loss criterion respectively.

		CIFAR-10				NUS-WIDE			
Category	Method	12bits	24bits	32bits	48bits	12bits	24bits	32bits	48bits
Traditional	FastH [20]	0.305	0.349	0.369	0.384	0.621	0.650	0.665	0.687
	SDH [21]	0.285	0.329	0.341	0.356	0.568	0.600	0.608	0.637
	KSH [22]	0.303	0.337	0.346	0.356	0.556	0.572	0.581	0.588
Deep Hashing	DQN [23]	0.554	0.558	0.564	0.580	0.768	0.776	0.783	0.792
	NINH [24]	0.552	0.566	0.558	0.581	0.674	0.697	0.713	0.715
	CNNH [25]	0.439	0.476	0.472	0.489	0.611	0.618	0.625	0.608
	DHN [26]	0.555	0.594	0.603	0.621	0.708	0.735	0.748	0.758
	CIBHash [27]	0.410	0.441	0.458	0.467	0.785	0.809	0.815	0.819
DHWP	Baseline (DPSH) [2]	0.678	0.718	0.728	0.735	0.785	0.807	0.798	0.814
	DHWP-Loss	**0.730**	0.723	0.723	0.742	0.792	0.809	0.814	0.819
	DHWP-MAP	0.706	0.726	0.730	**0.752**	0.790	0.807	0.814	0.818
	DHWP-Bal	0.713	**0.729**	**0.735**	0.749	**0.796**	**0.813**	**0.818**	**0.822**
	DHWP-Quant	0.718	0.719	0.718	0.741	0.790	0.809	0.815	0.819



(a) CIFAR-10 (b) NUS-WIDE

(c) CIFAR-10 (d) NUS-WIDE

Fig. 2. (a) and (b) are the Top N precision curves with code length 12 on two datasets. (c) and (d) are the precision-recall curves with code length 12 on two datasets.

equal to 12 and the MAP is about 1% to 2% higher when the length of the hash code is 48. For more difficult tasks on the multi-label dataset NUS-WIDE, the improvement of MAP is limited when the length of the hash code is 48. But when the hash code is relatively short, we can still see a significant improvement. Among four different pruning criteria, DHWP-Bal achieves better performance than other pruning strategies mostly, which shows that bit balance has an important effect on the deep supervised hashing. In practice, we can evaluate four different pruning criteria in the validation set and then make use of one of four different pruning criteria or mixed pruning criteria (e.g., selecting intersection or union set of pruning bits under different criteria). Overall, weight pruning can improve the performance of the baselines. In other words, we can use a shorter hash code to get a comparable search accuracy, thereby greatly improving the search speed. The effectiveness of our model is based on the effectiveness of pruning methods of deep neural networks.

3.3. Ablation Study

We consider the strength of fine-tuning and denote the pruned model without fine-tuning from the last pruned model as DHWP-inter. We compare the performance of DWHP-Inter and DWHP-Bal to show the effect of fine-tuning. The results are shown in Table 2. It can be found that DWHP achieves better performance consistently, which implies that fine-tuning is essential for our model. Besides, DHWP-Inter outperforms the baseline by 1% in average MAP for different hash code lengths, which demonstrates the effectiveness of pruning strategy compared with learning from the skeleton.

Table 2. Ablation study for DHWP

Method	12bits	24bits	32bits	48bits
Baseline	0.678	0.718	0.728	0.735
DHWP-Inter	0.701	0.720	0.730	0.745
DHWP-Bal	0.713	0.729	0.735	0.749

4. CONCLUSION

In this paper, we propose a novel framework named DHWP. DHWP starts from a deep supervised model and introduces a mask layer for iterative weight pruning. Numeric experiments demonstrate that DHWP can be applied to deep supervised methods and improve the performance of image retrieval, especially for short hash codes.

5. REFERENCES

[1] Xiao Luo, Chong Chen, Huasong Zhong, Hao Zhang, Minghua Deng, Jianqiang Huang, and Xiansheng Hua, "A survey on deep hashing methods," *arXiv preprint arXiv:2003.03369*, 2020.

[2] Wu-Jun Li, Sheng Wang, and Wang-Cheng Kang, "Feature learning based deep supervised hashing with pairwise labels," in *AAAI*, 2016.

[3] Xiao Luo, Yuhang Guo, Zeyu Ma, Huasong Zhong, Tao Li, Wei Ju, Chong Chen, and Minghua Deng, "Deep supervised hashing by classification for image retrieval," in *ICONIP*, 2021.

[4] Long-Kai Huang, Jianda Chen, and Sinno Jialin Pan, "Accelerate learning of deep hashing with gradient attention," in *ICCV*, 2019.

[5] Xiao Luo, Daqing Wu, Chong Chen, Jinwen Ma, and Minghua Deng, "Deep unsupervised hashing by global and local consistency," in *ICME*, 2021.

[6] Kevin Lin, Huei-Fang Yang, Jen-Hao Hsiao, and Chu-Song Chen, "Deep learning of binary hash codes for fast image retrieval," in *CVPR*, 2015.

[7] Chaoyou Fu, Liangchen Song, Xiang Wu, Guoli Wang, and Ran He, "Neurons merging layer: towards progressive redundancy reduction for deep supervised hashing," in *IJCAI*, 2019.

[8] Song Han, Jeff Pool, John Tran, and William Dally, "Learning both weights and connections for efficient neural network," in *NeurIPS*, 2015.

[9] Jonathan Frankle and Michael Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," *ICLR*, 2019.

[10] Reza Abbasi-Asl and Bin Yu, "Structural compression of convolutional neural networks based on greedy filter pruning," *arXiv preprint arXiv:1705.07356*, 2017.

[11] Xiao Luo, Weilai Chi, and Minghua Deng, "Deepprune: Learning efficient and interpretable convolutional networks through weight pruning for predicting dna-protein binding," *Frontiers in Genetics*, vol. 10, pp. 1145, 2019.

[12] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A Horowitz, and William J Dally, "Eie: efficient inference engine on compressed deep neural network," in *ISCA*, 2016.

[13] Song Han, Huizi Mao, and William J Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," in *NeurIPS*, 2012.

[15] Xingbo Liu, Xiushan Nie, and Yilong Yin, "Mutual linear regression-based discrete hashing," *arXiv preprint arXiv:1904.00744*, 2019.

[16] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell, "Rethinking the value of network pruning," *arXiv preprint arXiv:1810.05270*, 2018.

[17] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[18] Alex Krizhevsky, Geoffrey Hinton, et al., "Learning multiple layers of features from tiny images," 2009.

[19] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng, "Nus-wide: a real-world web image database from national university of singapore," in *Proceedings of the ACM international conference on image and video retrieval*, 2009.

[20] Guosheng Lin, Chunhua Shen, Qinfeng Shi, Anton Van den Hengel, and David Suter, "Fast supervised hashing with decision trees for high-dimensional data," in *CVPR*.

[21] Fumin Shen, Chunhua Shen, Wei Liu, and Heng Tao Shen, "Supervised discrete hashing," in *CVPR*, 2015.

[22] Wei Liu, Jun Wang, Rongrong Ji, Yu-Gang Jiang, and Shih-Fu Chang, "Supervised hashing with kernels," in *CVPR*, 2012.

[23] Yue Cao, Mingsheng Long, Jianmin Wang, Han Zhu, and Qingfu Wen, "Deep quantization network for efficient image retrieval," in *AAAI*, 2016.

[24] Hanjiang Lai, Yan Pan, Ye Liu, and Shuicheng Yan, "Simultaneous feature learning and hash coding with deep neural networks," in *CVPR*, 2015.

[25] Rongkai Xia, Yan Pan, Hanjiang Lai, Cong Liu, and Shuicheng Yan, "Supervised hashing for image retrieval via image representation learning," in *AAAI*, 2014.

[26] Han Zhu, Mingsheng Long, Jianmin Wang, and Yue Cao, "Deep hashing network for efficient similarity retrieval," in *AAAI*, 2016.

[27] Zexuan Qiu, Qinliang Su, Zijing Ou, Jianxing Yu, and Changyou Chen, "Unsupervised hashing with contrastive information bottleneck," in *IJCAI*, 2021.