Automatic Channel Pruning by Searching with Structure Embedding for Hash Network

Zifan Liu[1], **Yuan Cao***[1], **Yifan Sun**[1], **Yanwei Yu***[1], **Heng Qi**[2]

[1]School of Computer Science and Technology, Ocean University of China, China
[2]School of Computer Science and Technology, Dalian University of Technology, China
lzf9793@stu.ouc.edu.cn, cy8661@ouc.edu.cn, sunyifan5886@stu.ouc.edu.cn, yuyanwei@ouc.edu.cn, hengqi@dlut.edu.cn

Abstract

Deep hash networks are widely used in tasks such as large-scale image retrieval due to high search efficiency and low storage costs through binary hash codes. With the growing demand for deploying deep hash networks on resource-constrained devices, it is crucial to perform network compression on them, in which automatic pruning constitutes a priority option owing to efficacy maintenance. However, existing pruning methods are mostly designed for image classification, while hashing networks must generate compact binary codes, making each channel more sensitive to retrieval objectives. As a result, their performance often degrades when applied to image retrieval tasks. In this paper, we propose a novel Automatic Channel Pruning framework by Searching with Structure Embedding (ACP-SSE). To the best of our knowledge, this is the first study to explore pruning techniques for deep hash networks and the first automatic pruning method by searching based on network topology structure. Specifically, we first design a structure encoding model by Graph Convolutional Networks (GCNs) whose graph is constructed by hash network and nodes' features are initialized by pruning strategies. The model is trained by contrastive learning loss efficiently without accuracy supervision by fine-tuning pruned models. In addition, we introduce a dynamic pruning search space in consideration of the resource constraints. By converting the automatic channel pruning task into searching the pruned structure with effect similar to the unpruned structure, it enables the method to adapt to various network architectures. Finally, the optimal networks are selected from the candidate set according to their performance in specific downstream tasks. Extensive experiments demonstrate that ACP-SSE indeed works in the automatic channel pruning area, outperforming state-of-the-art baselines in hashing-based image retrieval, while maintaining competitive accuracy in image classification.

Code — https://github.com/caoyuan618/ACP-SSE

Introduction

Deep hashing (Qin et al. 2025; Cheng et al. 2025) has become a key tool in tasks such as image retrieval, with the goal of representing images as compact binary codes to enable efficient search and retrieval (Luo et al. 2023). However, deploying deep hashing models on resources-constrained devices (such as mobile phones and IoT devices) poses significant challenges. The computational and memory overhead of these models may hinder their performance in such environments, making model compression techniques crucial for practical deployment.

A large number of studies have focused on compression techniques for deep neural networks (Cong et al. 2020; Mi, Lei, and Gui 2013). Common methods include network pruning (Zhang et al. 2021; Wang, Li, and Wang 2021; Xu et al. 2020; Liu et al. 2020), quantization (Dong et al. 2019, 2023; Zhu et al. 2020; Wang et al. 2019), and knowledge distillation (Jin et al. 2021; Park et al. 2019; Gou et al. 2021). Among these, channel pruning has emerged as a key technique, effectively reducing computational costs by removing unimportant channels while maintaining model efficacy (He, Zhang, and Sun 2017; Luo, Wu, and Lin 2017). Early pruning methods rely on manual design or heuristic algorithms, which fail to optimally balance model compression and performance preservation. Hence, automated pruning methods have been proposed by leveraging optimization algorithms to autonomously identify and remove redundant components (Lin et al. 2020b; He et al. 2018; Sun, Cao, and Chen 2022). Unlike standard classification networks, hashing networks must generate compact binary encodings, which tightly aligns channel importance with retrieval targets. Consequently, pruning methods designed for classification models often fail to generalize to hashing networks, and their direct application to image retrieval tasks frequently yields suboptimal results.

To address the above challenges, we propose a novel Automatic Channel Pruning framework by Searching with Structure Embedding (ACP-SSE). Inspired by hashing retrieval, we correlate model performance with structural topology and adopt a search-based strategy, enabling the pruning process to naturally align with the retrieval mechanism. The core idea is that we believe network structure directly correlates with its feature extraction capability. By screening structures with effect similar to the unpruned one, the candidate set is formed and used for fine-tuning based on specific downstream tasks, thus endowing the method with high generality. To be specific, the overall architecture of a hash retrieval model is first represented as a graph structure. Each node in the graph corresponds to a convolutional

*Corresponding authors

layer, batch normalization (BN) layer, or pooling layer, and the edges represent the data flow relationships between these layers. Then, the channels corresponding to each convolutional layer are represented as features of the nodes, and the graph structure is processed by Graph Convolutional Networks (GCNs). The GCNs capture the global topological structure of the model and encodes its architectural features into a compact embedding vector. In order to guide the learning of channel importance in a meaningful way, we perform unsupervised contrastive learning between the embeddings of the original model and its L1-norm/random-based pruned variants as positive/negative samples, which encourages GCNs to generate structure-aware representations that reflect the performance potential of the architecture. Next, we design a pruning rate search space that defines multiple combinations of hierarchical pruning rates. A threshold is set based on the overall pruning rate of the model considering the resource constraints. It allows us to control the overall pruning rate, striking a balance between model size and accuracy. The size of this search space can be dynamically adjusted, allowing it to adapt to various downstream tasks and model complexities. Finally, based on the learned embeddings, we employ a search-based strategy that automatically determines the optimal pruning configuration without relying on manual rules or adjustments.

To the best of our knowledge, this is the first study to explore pruning techniques for deep hash networks and the first automatic pruning method by searching based on network topology structure. The main contributions of this work are summarized as follows:

- We propose a novel automatic channel pruning framework based on structure search for hash network, which enables topology-aware pruning and exhibits insensitivity to downstream tasks. A contrastive learning loss is constructed for training, leveraging positive and negative samples generated via L1-norm and random-based pruning strategies.
- We introduce a dynamic pruning search mechanism that optimizes the combination of pruning rates within a constrained space, allowing the pruning strategy to automatically adapt to various architectures and task complexities, thereby enhancing both flexibility and efficiency.
- Extensive experiments are conducted using multiple backbone models (ResNet, VGG16) with different hashing methods on benchmark datasets (CIFAR-10 and ImageNet100), demonstrating that our method outperforms other baselines in hashing-based image retrieval and is also competitive in image classification.

Method	Auto	Struct.	Search	Adapt.	General.
L1-Norm	✗	✗	✗	✗	✗
ThiNet	✗	✗	✗	✗	✗
Network Slimming	✗	✗	✗	✗	✗
HRank	✗	✗	✗	✗	✗
AMC	✓	✗	✓	–	✗
MetaPruning	✓	✗	✓	–	✗
ABCPruner	✓	✗	✓	✓	✗
AGMC	✓	✓	✓	–	✗
AutoSculpt	✓	✓	✓	✓	✗
ACP-SSE (Ours)	✓	✓	✓	✓	✓

Table 1: Comparison of representative pruning methods. Auto: Automated pruning; Struct.: Structural information; Search: Search-based strategy; Adapt.: Adaptability to different architectures; General.: Generalizability to different tasks.

Related Work

Deep Hashing

Deep hashing (Sun et al. 2023; Pu et al. 2025a,b) has been widely applied to large-scale image retrieval, where the goal is to map images to compact binary codes that can be efficiently compared for similarity. These models have shown promising results, significantly improving retrieval performance over traditional hashing methods. Deep hashing methods can be categorized into three main categories based on their learning strategies: pairwise-based, ranking-based, and center-based approaches. Pairwise-based methods (Li, Wang, and Kang 2015; Cao et al. 2018; Li et al. 2019; Wu et al. 2019) focus on learning binary encodings by exploiting pairwise relationships between images. These methods aim to minimize the distances between similar images and maximize the distances between dissimilar images in the Hamming space. Ranking-based methods (Wang, Shi, and Kitani 2016; Qin et al. 2023) optimize hash code learning by directly improving the ranking of images during retrieval. Center-based methods (Yuan et al. 2020; Fan et al. 2020; Wang et al. 2023) typically rely on clustering techniques, where hash codes are learned by approaching them to the centers of clusters. Despite their success in improving retrieval accuracy, deep hashing methods face significant challenges on resource-constrained devices due to large model size and high computational costs. Model compression techniques are crucial for reducing size and improving efficiency. Only one research ODH (He et al. 2024) has demonstrated the use of binary networks to compress deep hashing models, making them suitable for deployment in resource-constrained environments. However, network compression methods based on quantization such as ODH reduce the precision of network weights, which results in a significant degradation in retrieval accuracy.

Channel Pruning Channel pruning methods are primarily categorized into traditional and automated approaches. Traditional methods, such as L1-norm pruning (Li et al. 2016), rely on simple importance metrics like L1-norm of convolutional filters, assuming smaller weights contribute less to the output. More advanced approaches like ThiNet (Luo, Wu, and Lin 2017) and Network Slimming (Liu et al. 2017) consider the channel's impact on subsequent layers or apply sparsity regularization to batch normalization scaling factors. HRank (Lin et al. 2020a) evaluates channel importance by analyzing feature map rank, assuming lower-rank features carry less information. Although effective, these methods often rely on manually designed rules or heuristic algorithms, limiting their scalability across different architectures or tasks.

To address these limitations, automated pruning methods

are proposed. AMC (He et al. 2018) treats pruning as a reinforcement learning problem, where an agent determines the optimal compression strategy for each layer. MetaPruning (Liu et al. 2019) employs meta-learning to generate generalized pruning strategies, while ABCPruner (Lin et al. 2020b) utilizes an artificial bee colony algorithm to optimize pruning structures. Other approaches like AGMC (Yu, Mazaheri, and Jannesari 2021) combine graph neural networks with reinforcement learning to optimize pruning strategies. APIB (Guo et al. 2023) proposes pruning methods based on the information bottleneck principle to minimize heuristic dependencies. AutoSculpt (Jing et al. 2024) integrates graph neural networks with reinforcement learning to construct hardware-efficient pruning strategies. These automated approaches provide more scalable end-to-end solutions, significantly reducing reliance on heuristic methods.

As shown in Table 1, AGMC and AutoSculpt incorporate structural information through graph representations, AGMC utilizes graph neural networks to encode topology but relies on reinforcement learning for hierarchical decision making, while AutoSculpt focuses on hardware-friendly pattern regularity and applies graph-based reinforcement learning primarily to delay-aware pruning. And several other approaches, including MetaPruning, AMC, and ABCPruner, employ reinforcement or evolutionary strategies to introduce automated search mechanisms, typically operating at the layer or channel level. Our approach performs structure-aware pruning by searching in a learned embedding space that captures global architectural semantics. In terms of adaptability, only a few approaches, such as ABCPruner and AutoSculpt, exhibit moderate cross-architecture generalization capabilities. In contrast, our proposed ACP-SSE framework searches structural modeling and pruning strategies in a single pipeline, which satisfies all the performance mentioned above.

Methodology

In this section, we introduce the proposed ACP-SSE framework in detail. The general framework of the proposed ACP-SSE method is shown in Figure 1. ACP-SSE aims to efficiently prune convolutional neural networks by modeling the architectural structure explicitly and guiding the pruning process through a GCNs-based encoder trained via unsupervised contrastive learning. The goal is to automatically determine effective layer-wise pruning configurations.

Problem Definition

Given a deep hashing model $\mathcal{F}$ composed of L layers, where each layer l has C_l output channels, our objective is to find a set of layer-wise pruning ratios $\mathbf{r} = [r_1, r_2, ..., r_L]$, where $r_l \in [0, 1)$, such that the pruned model $\mathcal{F}'$ satisfies a desired global pruning ratio while maintaining competitive retrieval accuracy. The global pruning ratio $R(\mathbf{r})$ is defined as

$$R(\mathbf{r}) = \frac{\sum_{l=1}^{L} r_l \cdot C_l}{\sum_{l=1}^{L} C_l}. \tag{1}$$

where $R(\mathbf{r})$ represents the fraction of retained channels after pruning, and C_l is the number of channels in layer l. We enforce the constraint $R(\mathbf{r}) \geq \rho$, where ρ is a user-defined threshold that controls the desired level of compression in consideration of the resource constraints.

The challenge lies in identifying the optimal set of pruning ratios $\mathbf{r}$ from a high-dimensional, discrete search space that balances compression efficiency and retrieval accuracy, without requiring manual tuning or task-specific heuristics. More formally, we aim to solve the following optimization problem:

$$\min_{\mathbf{r}} \mathcal{L}(\mathbf{r}) \quad \text{subject to} \quad R(\mathbf{r}) \geq \rho, \tag{2}$$

where $\mathcal{L}(\mathbf{r})$ is the retrieval loss (e.g., Hamming distance between the predicted and true hash codes) of the pruned model $\mathcal{F}'$, which should be minimized while adhering to the pruning constraint. Thus, the problem involves searching for the optimal pruning ratio vector $\mathbf{r}$ that reduces the model size while preserving retrieval performance, without relying on manually defined pruning rules or heuristics.

Graph-Based Network Representation

To enable structure-aware pruning, we represent a neural network architecture as a Directed Acyclic Graph (DAG) $G = (V, E)$, where each node $v_i \in V$ denotes a specific computational layer, and each directed edge $e_{ij} \in E$ encodes the information flow from layer i to layer j. This formulation captures both sequential layer relationships and complex topological dependencies, such as skip connections, branching, and merging operations, which are common in modern architectures (e.g., ResNet). For standard feedforward layers, we connect each layer to its immediate successor. In the case of non-sequential modules such as residual blocks, we introduce edges that reflect long-range dependencies to preserve functional pathways in the graph. As a result, the graph topology explicitly models both shallow and deep hierarchical information in the original network.

To reflect the pruning status of each layer, we associate each node v_i with a binary feature vector $x_i \in \{0, 1\}^d$, where d is the maximum number of output channels among all layers:

$$d = \max_{l \in \{1,2,...,L\}} C_l. \tag{3}$$

The feature vector x_i represents the channel-wise mask for layer i: if the j-th channel is retained after pruning, then $x_i[j] = 1$; otherwise, $x_i[j] = -1$. Formally, for a given pruning vector $\mathbf{r}$ and corresponding preserved channel set $\mathcal{P}_i$ for layer i:

$$x_i[j] = \begin{cases} 1, & \text{if } j \in \mathcal{P}_i; \\ -1, & \text{otherwise.} \end{cases} \tag{4}$$

To handle layers with fewer than d channels, we zero-pad the feature vector to maintain consistent input dimensions across nodes. For Batch Normalization layers, we assign them the same feature vector as their preceding convolutional layer to preserve semantic coherence. Fully Connected (FC) layers are not subject to pruning in our setting and are therefore assigned an all-one vector: $x_i = \mathbf{1}_d$. This graph formulation encodes both the architectural topology

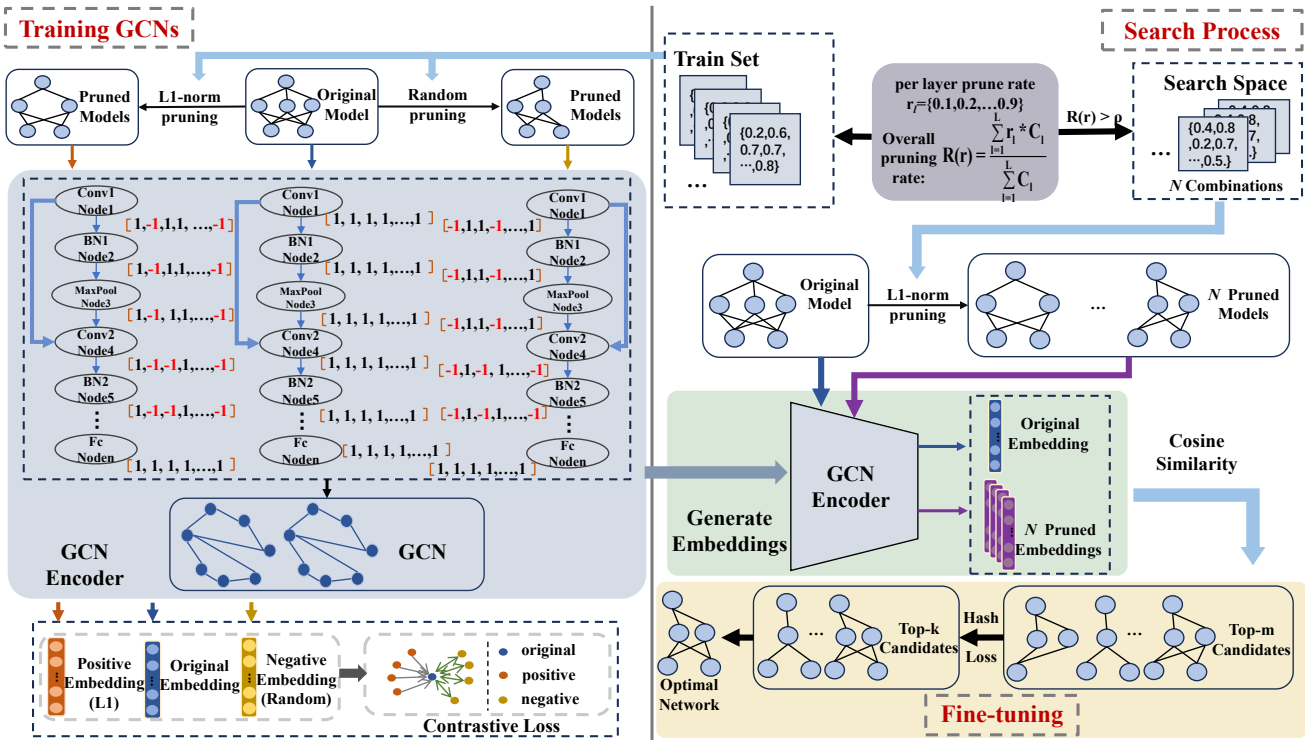

Figure 1: The overall framework of the proposed ACP-SSE method. Left: GCNs training. Pruned configurations are converted into DAGs with binary node features. The GCN encoder is trained via contrastive learning. Right: Structure-aware pruning. A large candidate pool is generated under a global pruning constraint. GCN embeddings are computed and compared with the original model via cosine similarity. Fine-tuning the candidate model to get the final pruned model.

and the fine-grained channel pruning state of the network. Such a representation enables the GCNs to jointly learn from the hierarchical structure and the pruning configuration, making it possible to reason about the global impact of pruning decisions at the architecture level.

Structured Pruning Dataset Generation

To train the GCN encoder to learn structure-aware pruning strategies, we construct two sets of pruning network configurations: one set for training the GCN encoder to improve its encoding capabilities, and the other set for searching during GCN encoder inference. Each set of configurations corresponds to a specific pruning strategy applied to all prunable layers, thereby generating a unique network architecture and its corresponding graph representation.

Let the original network contain L prunable layers, each with C_l output channels for $l = \{1, 2, \ldots, L\}$. We define a pruning vector $\mathbf{r} = [r_1, r_2, \ldots, r_L]$, where $r_l \in [0, 1)$ denotes the proportion of channels to prune in layer l. The remaining number of channels in layer l is computed as

$$C_l^{\text{pruned}} = \lfloor (1 - r_l) \cdot C_l \rfloor . \tag{5}$$

To keep the search space tractable while retaining sufficient expressiveness, we discretize each r_l into a fixed candidate set:

$$r_l \in \mathcal{R} = \{0, 0.1, 0.2, \ldots, 0.9\}. \tag{6}$$

Thus, the total number of possible pruning configurations for an L-layer network is $|\mathcal{R}|^L$.

For the GCN training stage, our goal is to enable the encoder to capture structural differences and their impact on model behavior. To this end, we randomly sample N pruning vectors $\{\mathbf{r}^{(1)}, \ldots, \mathbf{r}^{(N)}\}$ from the complete configuration space without imposing any pruning rate constraints. This strategy ensures high structural diversity in the training samples, enabling the GCN to learn robust and generalizable structure-aware embeddings.

All candidate solutions in the restricted set are evaluated using a structure-aware GCN encoder, which computes similarity scores via embeddings to guide pruning decisions. Each pruning configuration is represented as a graph $G^{(i)}$ with binary node features, as described in the graph construction section. The GCN is optimized via contrastive learning using the training configuration, while the restricted candidate set is leveraged during inference to select the optimal pruning strategy.

GCNs-Based Contrastive Embedding Learning

After constructing a graph representation of the pruned network and generating a pruning-rate search space, we train a graph encoder $\mathcal{G}_\theta$ based on GCNs to map each network graph into a low-dimensional embedding space that captures both the topological structure and specific pruning configurations, enabling the encoder to distinguish structural dif-

ferences and place well-pruned networks closer to the original model. Since precise pruning model performance labels cannot be obtained during encoder training, we adopt a contrastive learning framework: each iteration generates a triplet consisting of an anchor example, a positive example and a negative example: anchors are randomly drawn from the training set, positive examples are produced using an L1-norm-based pruning strategy that preserves information-rich channels and maintains structural integrity, and negative examples are generated by random pruning, which often disrupts critical structural dependencies. This design is motivated by the empirical finding that L1-norm based pruning consistently outperforms random pruning, creating a performance gap that guides the encoder to differentiate high-quality from suboptimal pruning configurations through their graph representations.

Given a batch of N anchor-contrast pairs, we compute the cosine similarity between each anchor embedding z_1 and its contrast embedding z_2, scaled by a temperature parameter ρ, as follows:

$$\text{sim}(z_1, z_2) = \frac{z_1 \cdot z_2^\top}{\tau}, \tag{7}$$

where τ is a temperature hyperparameter that controls the concentration level of the similarity distribution. For each anchor–positive pair $(z_{\text{anc}}^{(i)}, z_{\text{pos}}^{(i)})$, we denote $a = \max_j \text{sim}(z_{\text{anc}}^{(i)}, z_{\text{neg}}^{(j)})$ as the maximum similarity to any negative sample in the batch. The total contrastive loss is:

$$\mathcal{L}_{\text{CL}} = -\frac{1}{N}\sum_{i=1}^{N} \log \left(\frac{\exp\left(\text{sim}(z_{\text{anc}}^{(i)}, z_{\text{pos}}^{(i)}) - a\right)}{\sum_{j=1}^{N} \exp\left(\text{sim}(z_{\text{anc}}^{(i)}, z_{\text{neg}}^{(j)}) - a\right) + \epsilon} \right). \tag{8}$$

Here, a is a normalization constant defined as the maximum similarity between the anchor and all negative examples, which is subtracted from both the numerator and denominator to stabilize exponentiation and prevent numerical overflow. ϵ is a small constant added to avoid division by zero. This formulation encourages the encoder to assign higher similarity to the anchor–positive pair while pushing apart all negatives in the batch, enabling structure-aware embedding learning through contrastive objectives. This contrastive learning strategy guides the encoder to differentiate pruning configurations that preserve the structural integrity of the original network from those that disrupt it. Notably, this supervision does not require full retraining or inference. Instead, it enables the encoder to build a structure-aware embedding space that reflects the quality of pruning decisions, which can later be used for efficient pruning strategy search.

Pruning Strategy Selection and Final Optimization

Once the GCN encoder $\mathcal{G}_\theta$ is trained, it is used to evaluate the quality of candidate pruning configurations without requiring full model retraining. Specifically, we first generate a large pool of candidate pruning vectors $\{\mathbf{r}^{(j)}\}$, where each $\mathbf{r}^{(j)}$ satisfies the global pruning constraint $R(\mathbf{r}^{(j)}) \geq \rho$. This ensures that all candidates provide a meaningful level of compression while preserving model structure.

Each pruning vector $\mathbf{r}^{(j)}$ is then transformed into a feature graph $G^{(j)}$. The trained encoder $\mathcal{G}_\theta$ maps each graph into an embedding $z^{(j)}$, which represents the structural characteristics of the corresponding pruned network. To assess how well a candidate preserves the original model's structure, we compute its cosine similarity with the embedding of the unpruned model z_{orig}:

$$s^{(j)} = \cos(z^{(j)}, z_{\text{orig}}) = \frac{z^{(j)} \cdot z_{\text{orig}}}{\|z^{(j)}\| \cdot \|z_{\text{orig}}\|}. \tag{9}$$

A higher similarity score $s^{(j)}$ indicates greater alignment with the original architecture, and is thus correlated with better expected performance.

We then rank all candidates by similarity and select the top-m configurations. These candidates undergo lightweight fine-tuning to obtain an estimate of their actual performance. Based on the performance of the validation model, we further select the top-k candidates for full retraining until convergence. The final pruned model is selected as the one with the highest test performance among the k fully retrained candidates. This two-stage pruning strategy—first guided by structure-aware embeddings and then refined through empirical evaluation—enables effective selection of high-quality pruned networks.

Experiments

Datasets

We evaluate ACP-SSE on two standard image retrieval benchmarks: CIFAR-10 (Krizhevsky, Hinton et al. 2009) and ImageNet100 (Deng et al. 2009). CIFAR-10 comprises 10 classes with 60,000 images; following common deep hashing protocols, we randomly sample 100 images per class as queries, and use the remaining 50,000 images as the training database. ImageNet100 is a 100-class subset of ImageNet, where all images in these categories form the retrieval database. Validation images are used as queries, and we further randomly select 13,000 database images for training.

Baselines and Training Details

To evaluate the effectiveness of ACP-SSE in compressing deep hashing models, we apply our pruning framework to a set of representative hashing methods, including both pairwise, ranking-based, and center-based approaches, i.e., DCH (Cao et al. 2018), DFH (Li et al. 2019), DSHSD (Wu et al. 2019), DTSH (Wang, Shi, and Kitani 2016), DNSH (Qin et al. 2023), CSQ (Yuan et al. 2020), DPN (Fan et al. 2020), MDSH (Wang et al. 2023). We compare ACP-SSE with both generic and task-adapted channel pruning methods: HRank (Lin et al. 2020a), ABCPruner (Lin et al. 2020b), AutoSculpt(Jing et al. 2024), OTOv2 (Chen et al. 2023) and ATO (Wu et al. 2024).

The proposed model is implemented in PyTorch and experiments are conducted on three NVIDIA RTX 3090 GPUs. For each backbone, we construct a pruning-network training set by sampling configurations across sparsity levels and encoding them as binary graphs that mark pruned or retained

Backbone	Method	DCH	DFH	DSHSD	DTSH	DNSH	CSQ	DPN	MDSH
VGG-16	base	0.8882	0.8713	0.9112	0.8542	0.8061	0.9311	0.9019	0.8987
	ABCPruner	0.8764	0.8639	0.9115	0.8419	0.7814	0.9077	0.8814	0.8853
	AutoSculpt	0.8841	0.8769	0.9101	0.8318	0.7980	0.9118	0.9015	0.8870
	ACP-SSE (Ours)	**0.8854**	**0.8804**	**0.9137**	**0.8420**	**0.7992**	**0.9146**	**0.9018**	**0.8911**
ResNet-50	base	0.8046	0.8253	0.7787	0.8436	0.7972	0.8522	0.8243	0.8193
	ABCPruner	0.8013	0.8082	0.7593	**0.8394**	0.7767	0.8248	0.8077	0.7774
	AutoSculpt	0.8164	0.7982	0.7908	0.8362	0.7820	0.8464	0.7965	0.7975
	ACP-SSE (Ours)	**0.8215**	**0.8102**	**0.7915**	0.8387	**0.7875**	**0.8521**	**0.8146**	**0.8079**

Table 2: Comparison of mAP@all results using different hash models and pruning methods with VGG-16 and ResNet-50 in the image retrieval task on CIFAR-10.

Model	Method	DCH	DFH	DSHSD	DTSH	DNSH	CSQ	DPN	MDSH
ResNet-50	base	0.7030	0.7346	0.7130	0.6823	0.7768	0.7810	0.7073	0.7407
	ABCPruner	0.6433	0.6702	0.6697	**0.6543**	0.7386	0.7369	0.6467	0.6876
	AutoSculpt	0.6504	0.6687	0.6726	0.6511	**0.7401**	0.7405	0.6349	0.6842
	ACP-SSE (Ours)	**0.6608**	**0.6742**	**0.6781**	0.6432	0.7397	**0.7422**	**0.6468**	**0.6913**

Table 3: Comparison of mAP@all results using different hash models and pruning methods with ResNet-50 in the image retrieval task on ImageNet100.

channels. The GCN-based structure encoder is trained with contrastive learning for 10–20 epochs (batch size 32, learning rate 0.01, Adam, $\tau = 0.07$). Unless otherwise specified, hashing uses 64-bit codes. For ResNet-50, we generate 50k configurations and evaluate 1M candidates, keeping those with $< 50\%$ pruning, selecting the top-50 by structural similarity and fine-tuning the top-10. For VGG-16, we use 10k configurations and evaluate 50k candidates under a 60% pruning threshold. For ResNet-18, we similarly use 50k configurations and evaluate 100k candidates under a 70% threshold, selecting the top-150 before fine-tuning the top-10 models.

Results on Deep Hashing

We evaluate the proposed ACP-SSE framework on deep hashing models using two benchmark datasets: CIFAR-10 and ImageNet100. All experiments are conducted using 64-bit hash codes, and we report the retrieval performance in terms of mAP@all. To validate the compression effectiveness, we compare the performance of each hashing model before and after pruning.

Table 2 summarizes the results on CIFAR-10. Compared to other pruning methods such as ABCPruner and AutoSculpt, ACP-SSE consistently achieves superior or comparable mAP@all. In most cases, ACP-SSE preserves or improves retrieval accuracy compared to the original unpruned model. This demonstrates the effectiveness of structure-aware pruning in deep hashing, especially in resource-constrained scenarios. As shown in Table 3, ACP-SSE maintains competitive performance across various deep hashing models. Even under large-scale settings, it demonstrates strong robustness and retrieval accuracy, further validating the practicality of our pruning framework for real-world deployment scenarios. However, we note that the accuracy improvement in DTSH is relatively limited. This may be due to the fact that sort-based targets in DTSH are more sensitive to channel pruning, which disturbs the fine-grained pairwise distance structure required for accurate similarity sorting.

Results on Image Classification

To further evaluate the generalization of ACP-SSE beyond retrieval tasks, we apply it to the standard image classification benchmark on CIFAR-10 using two widely adopted backbone networks: VGG-16 and ResNet-18. Table 4 summarizes the results. On VGG-16, ACP-SSE achieves a 92.76% Top-1 accuracy, while reducing 82.23% of FLOPs and pruning 92.12% of the parameters—demonstrating an excellent balance between efficiency and accuracy. Compared to ABCPruner and HRank, ACP-SSE provides higher compression with only marginal loss in accuracy. On ResNet-18, ACP-SSE achieves a Top-1 accuracy of 93.41%, outperforming OTOv2 and ATO in both accuracy and parameter compression. It reduces 84.99% of FLOPs and 92.67% of parameters. These results confirm that ACP-SSE

Model	Method	Top-1 Accuracy	FLOPs	Pruned FLOPs	Parameters	Pruned Parameters
VGG-16	Base	92.60%	314.31M	0.00%	14.73M	0.00%
	GAL-0.05	90.86%	189.49M	39.71%	3.36M	77.19%
	Hrank	91.26%	108.61M	65.44%	2.64M	82.08%
	ABCPruner-80%	**93.08%**	82.81M	73.65%	1.67M	88.68%
	ACP-SSE (ours)	92.76%	**55.86M**	**82.23%**	**1.16M**	**92.12%**
ResNet-18	Base	92.32%	557.89M	0.00%	11.17M	0.00%
	OTOv2	92.86%	113.25M	79.70%	-	-
	ATO	**94.51%**	112.69M	79.80%	-	-
	ACP-SSE (ours)	93.41%	**83.76M**	**84.99%**	**0.82M**	**92.67%**

Table 4: Comparison of performance results using different pruning methods with VGG-16 and ResNet-18 in the image classification task on CIFAR-10.

Method	DCH	DTSH	CSQ
ACP-SSE-1	0.7623	0.7832	0.8041
ACP-SSE-2	0.7748	0.7887	0.8175
ACP-SSE	**0.8215**	**0.8387**	**0.8521**

Table 5: Ablation results of ACP-SSE using different hash models with ResNet-50 on CIFAR-10.

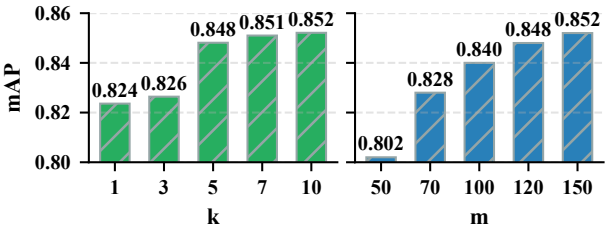

Figure 2: Effect of k and m values on pruning performance using CSQ hashing model with ResNet-50 on CIFAR-10.

not only generalizes to classification tasks but also achieves competitive or superior performance compared to strong pruning baselines.

Ablation Study

To evaluate the contribution of each component in the ACP-SSE framework for hash retrieval, we conduct controlled ablation studies on ResNet-18 using CIFAR-10 under 64-bit code settings. Specifically, we analyze the effect of two core components: unsupervised contrastive learning and the top-k refinement strategy. ACP-SSE-1 replaces the contrastive learning used to train the GCN encoder with a supervised regression objective, where a linear layer predicts the mAP of each candidate configuration. ACP-SSE-2 removes the top-k refinement stage and directly selects the highest-ranking configuration based on cosine similarity. As shown in Table 5, both ablated variants lead to a noticeable drop in retrieval performance across different hash models. The complete ACP-SSE model consistently outperforms both variants, demonstrating the benefit of contrastive learning and the top-k refinement strategy in hash retrieval pruning.

Hyperparameter Analyses

We investigate the impact of two key hyperparameters in the ACP-SSE framework: the number of top candidates retained after similarity-based ranking (k) and the size of the pruning candidate pool (m). As shown in Figure 2, when $m = 50$ is fixed, increasing k from 1 to 10 consistently improves retrieval performance. This indicates that exploring a broader subset of high-ranking candidates enables the framework to mitigate ranking noise and optimize pruning decisions. Similarly, when k is fixed at 10 and m is varied, we observe that a larger candidate pool leads to higher retrieval accuracy due to increased diversity. However, when m exceeds 50, the improvement effect gradually saturates. It is also worth noting that increasing either k or m leads to a larger number of candidates requiring fine-tuning, which results in higher computational costs. To balance retrieval performance and efficiency, we select moderate values (e.g., $m = 50$, $k = 10$) as a practical trade-off in our experiments.

Conclusion

In this paper, we propose a novel Automatic Channel Pruning By Searching with Structure Embedding method called ACP-SSE. To our knowledge, this is the first work to implement pruning on deep hash and the first attempt to transform automatic pruning into searching based on network embeddings. By modeling networks as graphs and training a GCN encoder via unsupervised contrastive learning, ACP-SSE captures global topology and learns pruning policies without performance labels. A task-adaptive search space and embedding-based evaluation further enable efficient configuration selection. Experimental results show its effectiveness in both image retrieval and image classification tasks.

Acknowledgements

This work is partially supported by the National Science Foundation of China under grant Nos. 62202438, 62176243; the Natural Science Foundation of Shandong Province Grant No. ZR2024MF128.

References

Cao, Y.; Long, M.; Liu, B.; and Wang, J. 2018. Deep cauchy hashing for hamming space retrieval. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1229–1237.

Chen, T.; Liang, L.; Ding, T.; Zhu, Z.; and Zharkov, I. 2023. Otov2: Automatic, generic, user-friendly. *arXiv preprint arXiv:2303.06862*.

Cheng, K.; Qin, Q.; Zhang, W.; Huang, L.; and Nie, J. 2025. Deep Probabilistic Binary Embedding via Learning Reliable Uncertainty for Cross-Modal Retrieval. In *Proceedings of the ACM international conference on multimedia*, 6393–6402.

Cong, X.; Gui, J.; Miao, K.-C.; Zhang, J.; Wang, B.; and Chen, P. 2020. Discrete haze level dehazing network. In *Proceedings of the ACM international conference on multimedia*, 1828–1836.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *Proceedings of IEEE conference on computer vision and pattern recognition*, 248–255. Ieee.

Dong, P.; Li, L.; Wei, Z.; Niu, X.; Tian, Z.; and Pan, H. 2023. Emq: Evolving training-free proxies for automated mixed precision quantization. In *Proceedings of the IEEE international conference on computer vision*, 17076–17086.

Dong, Z.; Yao, Z.; Gholami, A.; Mahoney, M. W.; and Keutzer, K. 2019. Hawq: Hessian aware quantization of neural networks with mixed-precision. In *Proceedings of the IEEE international conference on computer vision*, 293–302.

Fan, L.; Ng, K. W.; Ju, C.; Zhang, T.; and Chan, C. S. 2020. Deep polarized network for supervised learning of accurate binary hashing codes. In *IJCAI*, volume 825.

Gou, J.; Yu, B.; Maybank, S. J.; and Tao, D. 2021. Knowledge distillation: A survey. *International journal of computer vision*, 129(6): 1789–1819.

Guo, S.; Zhang, L.; Zheng, X.; Wang, Y.; Li, Y.; Chao, F.; Wu, C.; Zhang, S.; and Ji, R. 2023. Automatic network pruning via hilbert-schmidt independence criterion lasso under information bottleneck principle. In *Proceedings of the IEEE international conference on computer vision*, 17458–17469.

He, L.; Huang, Z.; Liu, C.; Li, R.; Wu, R.; Liu, Q.; and Chen, E. 2024. One-bit deep hashing: Towards resource-efficient hashing model with binary neural network. In *Proceedings of the ACM international conference on multimedia*, 7162–7171.

He, Y.; Lin, J.; Liu, Z.; Wang, H.; Li, L.-J.; and Han, S. 2018. Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European conference on computer vision*, 784–800.

He, Y.; Zhang, X.; and Sun, J. 2017. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE international conference on computer vision*, 1389–1397.

Jin, Q.; Ren, J.; Woodford, O. J.; Wang, J.; Yuan, G.; Wang, Y.; and Tulyakov, S. 2021. Teachers do more than teach: Compressing image-to-image models. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 13600–13611.

Jing, L.; Qi, J.; Dong, J.; and Yu, Y. 2024. AutoSculpt: A pattern-based model auto-pruning framework using reinforcement Learning and Graph Learning. *arXiv preprint arXiv:2412.18091*.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images.

Li, H.; Kadav, A.; Durdanovic, I.; Samet, H.; and Graf, H. 2016. Pruning filters for efficient convnets. *Cornell University - arXiv*.

Li, W.-J.; Wang, S.; and Kang, W.-C. 2015. Feature learning based deep supervised hashing with pairwise labels. *arXiv preprint arXiv:1511.03855*.

Li, Y.; Pei, W.; van Gemert, J.; et al. 2019. Push for quantization: Deep fisher hashing. *arXiv preprint arXiv:1909.00206*.

Lin, M.; Ji, R.; Wang, Y.; Zhang, Y.; Zhang, B.; Tian, Y.; and Shao, L. 2020a. Hrank: Filter pruning using high-rank feature map. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1529–1538.

Lin, M.; Ji, R.; Zhang, Y.; Zhang, B.; Wu, Y.; and Tian, Y. 2020b. Channel pruning via automatic structure search. *ArXiv preprint arXiv:2001.08565*.

Liu, N.; Ma, X.; Xu, Z.; Wang, Y.; Tang, J.; and Ye, J. 2020. Autocompress: An automatic dnn structured pruning framework for ultra-high compression rates. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 4876–4883.

Liu, Z.; Li, J.; Shen, Z.; Huang, G.; Yan, S.; and Zhang, C. 2017. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE international conference on computer vision*, 2736–2744.

Liu, Z.; Mu, H.; Zhang, X.; Guo, Z.; Yang, X.; Cheng, K.-T.; and Sun, J. 2019. Metapruning: Meta learning for automatic neural network channel pruning. In *Proceedings of the IEEE international conference on computer vision*, 3296–3305.

Luo, J.-H.; Wu, J.; and Lin, W. 2017. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, 5058–5066.

Luo, X.; Wang, H.; Wu, D.; Chen, C.; Deng, M.; Huang, J.; and Hua, X.-S. 2023. A survey on deep hashing methods. *ACM transactions on knowledge discovery from data*, 17(1): 1–50.

Mi, J.-X.; Lei, D.; and Gui, J. 2013. A novel method for recognizing face with partial occlusion via sparse representation. *Optik*, 124(24): 6786–6789.

Park, W.; Kim, D.; Lu, Y.; and Cho, M. 2019. Relational knowledge distillation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3967–3976.

Pu, R.; Qin, Y.; Song, X.; Peng, D.; Ren, Z.; and Sun, Y. 2025a. SHE: Streaming-media Hashing Retrieval. In *Proceedings of the international conference on machine learning*.

Pu, R.; Sun, Y.; Qin, Y.; Ren, Z.; Song, X.; Zheng, H.; and Peng, D. 2025b. Robust Self-Paced Hashing for Cross-Modal Retrieval with Noisy Labels. In *Proceedings of the AAAI conference on artificial intelligence*, volume 39, 19969–19977.

Qin, Q.; Huo, Y.; Zhang, W.; Huang, L.; and Nie, J. 2025. Deep Discriminative Boundary Hashing for Cross-modal Retrieval. *IEEE transactions on circuits and systems for video technology*.

Qin, Q.; Xie, K.; Zhang, W.; Wang, C.; and Huang, L. 2023. Deep neighborhood structure-preserving hashing for large-scale image retrieval. *IEEE transactions on multimedia*, 26: 1881–1893.

Sun, Q.; Cao, S.; and Chen, Z. 2022. Filter pruning via automatic pruning rate search. In *Proceedings of the Asian conference on computer vision*, 4293–4309.

Sun, Y.; Ren, Z.; Hu, P.; Peng, D.; and Wang, X. 2023. Hierarchical consensus hashing for cross-modal retrieval. *IEEE transactions on multimedia*, 26: 824–836.

Wang, K.; Liu, Z.; Lin, Y.; Lin, J.; and Han, S. 2019. Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 8612–8620.

Wang, L.; Pan, Y.; Liu, C.; Lai, H.; Yin, J.; and Liu, Y. 2023. Deep hashing with minimal-distance-separated hash centers. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 23455–23464.

Wang, X.; Shi, Y.; and Kitani, K. M. 2016. Deep supervised hashing with triplet labels. In *Proceedings of Asian conference on computer vision*, 70–84. Springer.

Wang, Z.; Li, C.; and Wang, X. 2021. Convolutional neural network pruning with structural redundancy reduction. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 14913–14922.

Wu, L.; Ling, H.; Li, P.; Chen, J.; Fang, Y.; and Zhou, F. 2019. Deep supervised hashing based on stable distribution. *IEEE Access*, 7: 36489–36499.

Wu, X.; Gao, S.; Zhang, Z.; Li, Z.; Bao, R.; Zhang, Y.; Wang, X.; and Huang, H. 2024. Auto-train-once: Controller network guided automatic network pruning from scratch. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 16163–16173.

Xu, P.; Cao, J.; Shang, F.; Sun, W.; and Li, P. 2020. Layer pruning via fusible residual convolutional block for deep neural networks. *arXiv preprint arXiv:2011.14356*.

Yu, S.; Mazaheri, A.; and Jannesari, A. 2021. Auto graph encoder-decoder for neural network pruning. In *Proceedings of the IEEE international conference on computer vision*, 6362–6372.

Yuan, L.; Wang, T.; Zhang, X.; Tay, F. E.; Jie, Z.; Liu, W.; and Feng, J. 2020. Central similarity quantization for efficient image and video retrieval. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3083–3092.

Zhang, T.; Ye, S.; Feng, X.; Ma, X.; Zhang, K.; Li, Z.; Tang, J.; Liu, S.; Lin, X.; Liu, Y.; et al. 2021. Structadmm: Achieving ultrahigh efficiency in structured pruning for dnns. *IEEE transactions on neural networks and learning systems*, 33(5): 2259–2273.

Zhu, F.; Gong, R.; Yu, F.; Liu, X.; Wang, Y.; Li, Z.; Yang, X.; and Yan, J. 2020. Towards unified int8 training for convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1969–1979.