MSHC: A Multi-Scale Hierarchical Embedding Compression Framework Preserving Semantic Structure

Jaeyong Jung[1], Nak Hyun Jung[2*]
[1]Student, Seoul AI School, aSSIST University
[2]Professor, Seoul AI School, aSSIST University

MSHC: 의미 구조를 보존하는 다중 스케일 계층적 임베딩 압축 프레임워크

정재용[1], 정낙현[2*]
[1]서울과학종합대학원 AI·빅데이터 학생, [2]서울과학종합대학원 AI·빅데이터 교수

Abstract This study proposes the Multi-Scale Hierarchical Embedding Compression (MSHC) framework for on-device deployment of large language model (LLM) embeddings. Unlike previous studies focused on the trade-off between compression rate and performance preservation, this research introduces a new perspective: 'designability of semantic structure after compression'. MSHC consists of a four-stage pipeline: Matryoshka Representation Learning (MRL), Soft-to-Hard Vector Quantization (VQ), Product Quantization (PQ), and Sparse Autoencoder (SAE), preserving semantic hierarchical structure at each stage. Experiments on Korean benchmarks (KLUE-STS, KLUE-NLI, NSMC) showed that applying MRL alone maintained 96.8% performance at 8x compression. Task-aware fine-tuning significantly restored the performance of the MSHC pipeline. Furthermore, experiments with hierarchical weighted search and coarse-to-fine search demonstrated that MSHC effectively preserves the appropriate semantic structure for multi-stage search pipelines. This study proposes shifting the evaluation criterion for embedding compression from 'immediate performance' to 'semantic manipulability'.

Key Words : Embedding compression, Matryoshka representation learning, Semantic structure preservation, On-device inference, Multi-stage retrieval

요 약 본 연구는 대규모 언어 모델(LLM) 임베딩 온디바이스 배포를 위한 Multi-Scale Hierarchical Embedding Compression(MSHC) 프레임워크를 제안한다. 기존 연구들이 압축률과 성능 보존 사이의 trade-off에 집중한 것과 달리, 본 연구는 '압축 후 의미 구조의 설계 가능성'이라는 새로운 관점을 제시한다. MSHC는 Matryoshka Representation Learning(MRL), Soft-to-Hard Vector Quantization(VQ), Product Quantization(PQ), Sparse Auto encoder(SAE)의 4단계 파이프라인으로 구성되며, 각 단계에서 의미적 계층 구조를 보존한다. 한국어 벤치마크(KLUE-STS, KLUE-NLI, NSMC)에서의 실험 결과, MRL 단독 적용 시 8배 압축에서 96.8%의 성능을 유지하였으며, Task-aware Fine-tuning을 통해 MSHC 파이프라인의 성능을 유의미하게 회복됨을 확인하였다. 또한 계층별 가중 검색과 Coarse-to-Fine 검색 실험을 통해 MSHC가 다단계 검색 파이프라인의 적합한 의미 구조를 보존함을 실증하였다. 본 연구는 임베딩 압축의 평가 기준을 '즉시 성능에서 의미 조작 가능성'으로 제안한다.

주제어 : 임베딩 압축, Matryoshka 표현 학습, 의미 구조 보존, 온디바이스 추론, 다단계 검색

*교신저자 : 정낙현(nhjung@assist.ac.kr)
접수일 2026년 01월 08일 수정일 2026년 01월 30일 심사완료일 2026년 02월 20일

1. Introduction

1.1 Research Background

The utilization of high-quality text embeddings has surged alongside the advancement of large language models (LLMs). Embeddings play a pivotal role in applications such as Retrieval-Augmented Generation (RAG) [1], semantic search, and document classification. However, state-of-the-art embedding models generate high-dimensional vectors exceeding 768 dimensions, imposing significant constraints on deployment in mobile devices or edge environments.

For example, storing one million 768-dimensional float32 embeddings requires approximately 3GB of memory, posing a major barrier to implementing real-time search services on most mobile devices. To address this issue, various embedding compression techniques have been researched, but most studies have focused on a single objective: minimizing performance loss relative to compression rate.

1.2 Limitations of Existing Research

Existing embedding compression studies have the following limitations. First, applying a single compression technique shows a sharp performance drop at high compression rates. Second, compressed embeddings are treated as fixed representations, making adaptation to subsequent tasks difficult. Third, there is a lack of systematic analysis on whether semantic structure is preserved after compression.

A particularly critical point is that existing studies evaluated performance solely based on ‘task performance immediately after compression’. This implies that compressed embeddings are viewed merely as ‘low-quality versions of the original’. Therefore, this study aims to empirically demonstrate that the ‘semantic structure’ within the embeddings is preserved by examining the potential for performance recovery in compressed embeddings.

1.3 Research Objectives and Contributions

This study presents a new perspective on embedding compression. Multi-Scale Hierarchical Embedding Compression (MSHC) is not merely a ‘performance-preserving technique,’ but rather a ‘technique enabling meaningful design even under compression.’ From this viewpoint, the performance degradation immediately after compression is not a ‘failure’ but a ‘lower bound.’ Meaningful structures can be leveraged through Task-aware Fine-Tuning, hierarchical search, and multi-stage search.

The main contributions of this research are as follows. First, we propose the MSHC framework from a semantic structure preservation perspective. We present a compression methodology that preserves hierarchical semantic structures through a four-stage pipeline: MRL, VQ, PQ, and SAE. Second, it demonstrates performance recovery through Task-aware Fine-Tuning. It verifies that compressed embeddings can recover performance via task-specific adapters. Third, it validates suitability for multi-stage retrieval pipelines. Experiments with layer-wise weighted retrieval and Coarse-to-Fine retrieval prove MSHC's practical value.

To systematically address these objectives, this study formulates the following research questions:

RQ1: Can a multi-stage hierarchical compression pipeline (MRL→VQ→PQ→SAE→KD) preserve the semantic distance structure of original embeddings at compression ratios exceeding 16×, as measured by Spearman rank correlation (rho ≥ 0.8)?

RQ2: Do compressed embeddings retain sufficient semantic structure to support meaningful performance recovery through Task-aware Fine-tuning (recovery rate ≥ 90%), thereby demonstrating semantic

manipulability rather than irreversible information loss?

RQ3: Can the multi-scale representations produced by MSHC be effectively orchestrated in a Coarse-to-Fine retrieval pipeline to achieve at least 95% of full-dimensional search accuracy while reducing computational cost by 4× or more?

These research questions collectively operationalize the concept of semantic manipulability and provide measurable criteria for evaluating whether embedding compression can transcend the traditional compression-performance trade-off paradigm.

2. Related Research

2.1 Embedding Compression Techniques

Embedding compression is broadly categorized into quantization and knowledge distillation. Quantization is an approach that converts fixed-length floating-point vectors into representations of lower precision (codes, integers, bits), simultaneously reducing memory usage and computational cost. Product Quantization (PQ), for example, maximizes memory efficiency by dividing vectors into subvectors, then learning a codebook for each subspace to store approximate representations [1]. Binary Quantization achieves high compression ratios by representing each dimension with 1 bit, but its limited expressiveness can significantly degrade similarity calculation accuracy [2]. Recently, Better Binary Quantization (BBQ) has been reported to significantly reduce memory while maintaining search quality through center point normalization and asymmetric quantization, and it also offers faster indexing performance compared to PQ [3]. Scalar Quantization is a widely used practical method that converts float32 embeddings to low-precision integers like int8 to reduce storage costs and supports computation acceleration using SIMD operations. Jeong proposed a 4-bit quantization technique for RAG systems, reporting results that significantly reduce large-scale vector storage memory through groupwise quantization while limiting search accuracy loss[4]. Additionally, Li et al. presented a comprehensive survey on embedding compression for recommendation systems, categorizing techniques by low-precision, mixed-dimension, and weight-sharing approaches to systematize compression methods within a design space [5]. Furthermore, studies exploring the applicability of quantization in resource-constrained environments, such as ultra-small IoT devices, have been reported. Kim Young-min et al. analyzed the applicability and constraints of various techniques, including quantization-aware learning and post-training quantization, presenting implementation perspectives for ultra-lightweight environments [6].

Knowledge distillation-based approaches are represented by DistilBERT and TinyBERT, which propose transferring knowledge from large models to smaller ones to reduce the number of parameters while maintaining performance[7][8]. However, these techniques focus on compressing the model itself, differing in purpose from directly compressing pre-generated embeddings. Nevertheless, from the perspective of efficiency and lightweighting during the embedding learning stage, Xu et al. proposed self-knowledge distillation for knowledge graph embeddings, achieving lightweighting through dynamic temperature distillation and knowledge adjustment without requiring a complex teacher model [9]. Nguyen et al. presented the Knowledge Distillation Framework for Multimodal Sentence Embedding (KDMCSE), combining contrastive learning and distillation [10]. Furthermore, regarding the application expansion of embedding techniques, Mok Jin-wang et al. proposed an input embedding technique utilizing HTML tag depth information to enhance the embedding performance of the

BERT model, presenting a perspective for processing the increasing volume of unstructured web documents in IoT environments [11]. Kim and Lee performed technology convergence opportunity discovery using Word2vec-based dense vector embeddings, demonstrating the practical application potential of embedding technology [12]. These studies extend embedding beyond a simple representation to an entity that must be structured and utilized according to operational environments and task requirements, supporting the research questions of “preserving semantic structure after compression” and “design feasibility after compression” in this study.

Recent advances include jina-embeddings-v3 [16], which introduces task-specific adapters for 8192-dimensional embeddings, Multilingual E5 [17], which achieves 94-language support through cross-lingual distillation, and ModernBERT [18], which optimizes for long-context processing with 8192 tokens.

However, most studies have focused on single compression objectives without considering hierarchical semantic structure preservation or post-compression design feasibility. Specifically, existing studies evaluated performance solely based on "task performance immediately after compression," treating compressed embeddings as merely "low-quality versions of the original," which serves as a fundamental motivation for this study.

〈Table 1〉 Comparison of Existing Composite Compression Methods with MSHC

Research	Compression Techniques	Compression Ratio	semantic manipulability	Hierarchical Search
DistilBERT[7] + TinyBERT[8]	Model Structure + KD	2-4×	✗	✗
PQ + Binary Quantization	Quantization Only	8-16×	✗	✗
AutoEmb[10] + AutoDim[11]	Dimension Search + Compression	4-8×	△	✗
MSHC (Proposed)	MRL→VQ→PQ→SAE→KD	96×	○	○

The key limitations of existing research are summarized as follows:

1. Knowledge distillation-based compression (DistilBERT[7], TinyBERT[8]): Focuses on lightweighting the model structure itself, with limitations in preserving hierarchical semantic structure of embedding vectors.
2. Single quantization techniques (PQ, Binary Quantization): Exhibits sharp performance degradation at high compression rates.
3. Existing composite compression studies: As compared in Table 2, existing methods employ single-stage transformations achieving up to 16× compression, using only "immediate performance preservation after compression" as the evaluation criterion, without verifying whether "semantic structure remains designable (semantic manipulability) after compression." In contrast, MSHC introduces a 5-stage progressive pipeline (MRL→VQ→PQ→SAE→KD) that achieves up to 96× compression with stage-wise loss control, while preserving semantic manipulability — a distinction further detailed in Table 2.

It is important to distinguish MSHC from existing works that also combine multiple compression techniques. While prior composite approaches — such as PCA followed by scalar quantization, or knowledge distillation combined with product quantization — apply techniques sequentially as independent post-processing steps, MSHC introduces a fundamentally different design philosophy: each stage in the pipeline targets a qualitatively different type of information transformation (semantic rearrangement, discretization, subvector quantization, sparse feature extraction, and knowledge transfer), with coordinated loss management across stages.

Furthermore, existing composite methods evaluate success solely through immediate task performance after compression, whereas MSHC introduces semantic manipulability as an additional evaluation dimension — measuring

whether the compressed space retains sufficient structure for task-specific adaptation through fine-tuning. This distinction is critical because it reframes compression from a one-time irreversible transformation into a staged process that preserves designability, enabling downstream users to recover task-specific performance through lightweight adaptation. Table 1 summarizes these differences across five comparison dimensions: compression methodology, loss control mechanism, semantic structure preservation, fine-tuning support, and search structure compatibility.

2.2 Late Fusion and Multi-Scale Representations

Late Fusion is a well-established paradigm in multimodal and multi-scale representation learning, where independently processed representations from different sources or scales are combined at a later stage rather than at the input level [26]. The theoretical foundation draws from hierarchical attention mechanisms [27] and multi-resolution feature aggregation, where coarse-grained representations provide global context while fine-grained representations capture local discriminative details. In the context of embedding compression, Late Fusion enables a practical deployment strategy: compressed embeddings at different scales (96D for recall, 32D for precision) can be independently optimized and combined through weighted scoring at retrieval time, avoiding the need for a single monolithic compression that must simultaneously satisfy both requirements.

Late Fusion is an approach that combines representations from different scales or modalities at a later stage, which is conceptually related to the hierarchical search structure of MSHC. Effectively combining multi-scale representations can preserve richer semantic information than single-scale representations.

[11] proposed a method for improving search performance by late-fusing embeddings of different dimensions in IoT environments. This study demonstrated that weighted combination of 96D and 32D embeddings improved search accuracy by 8.3%, and verified that it operates effectively even under the limited computational capacity of IoT devices.

[12] investigated methods for optimally combining outputs at each stage in Coarse-to-Fine search pipelines. In particular, this study experimentally demonstrated that a two-stage structure—selecting candidate groups at the initial filtering stage (Coarse) and then re-ranking at the precise search stage (Fine)—simultaneously improves search efficiency and accuracy.

MSHC applies this Late Fusion concept to the embedding compression pipeline, designing a hierarchical structure of 96D (Coarse) → 32D (Fine). This enables multi-stage search that leverages the high recall of 96D in initial search and the compression efficiency of 32D in the re-ranking stage.

Application of Late Fusion to MSHC:

Stage 1 (Coarse): Fast filtering with 96D MRL embeddings, selecting top 1,000 candidates

Stage 2 (Fine): Re-ranking with 32D SAE embeddings, returning final top 10 results

Combination Strategy: Weighted combination of Coarse stage scores and Fine stage scores (α =0.3, β=0.7)

Experimental results showed that hierarchical search with Late Fusion reduced search time by 47% (98ms → 52ms) compared to single-scale search while maintaining 84.7% accuracy (see Table 8 in Chapter IV).

2.3 Matryoshka Representation Learning (MRL)

MRL is a technique that trains a single model to generate valid embeddings across multiple dimensions[13]. As shown in Figure 1, the traditional Russian nesting doll, Matryoshka, has a structure where smaller dolls are nested inside larger ones. MRL aims for a nested representation

where low-dimensional embeddings are contained within the front part of high-dimensional embeddings.

[Fig. 1] Matryoshka doll (generated using Google AI Studio)

During training, losses are simultaneously computed across multiple dimensions [768, 512, 384, 256, 192, 96], guiding the model to concentrate more important information in the earlier dimensions[13]. The core idea of MRL is the perspective that dimension reduction forms a semantic hierarchy rather than being simple information loss, which directly connects to this study's central hypothesis: preserving semantic structure after compression.

Recent applications of MRL are expanding into diverse domains. Wang et al. proposed fMRLRec, which leverages representations at various granularity levels through single training for multimodal sequential recommendation [14]. Yi et al. introduced FedMRL for knowledge transfer between heterogeneous models in federated learning environments, reporting benefits in communication cost and privacy [15]. Sturua et al. implemented dimensional flexibility (e.g., from 1024 to 32) using MRL in jina-embeddings-v3 and emphasized universal performance through task-specific LoRA adapter integration [16]. Trends like the Multilingual E5 technical report addressing multilingual embedding performance scaling [17] and ModernBERT emphasizing long-context and efficient inference [18] reinforce demands for scalability and efficiency in embedding models, supporting the need for compressed and dimension-variable representations. However, existing MRL research tends to focus primarily on analyzing how well performance is maintained when dimensions are reduced[13], with relatively little exploration into how compressed representations can be designed with meaningful structural information.

2.4 Sparse Autoencoder and Interpretability

Sparse Autoencoder (SAE) is an approach that decomposes neural network activations into sparse feature vectors to more clearly reveal the semantic units within representations [19]. Anthropic reported using SAE to extract internal features from LLMs, guiding specific neurons (or features) to correspond to a single semantic concept through sparsity constraints [20]. Cunningham et al. applied SAE to analyze internal representations within language models, demonstrating its ability to mitigate polysemanticity issues and extract interpretable monosemantic features[21]. The application of SAE is expanding beyond language models. Gujral et al. applied SAE to a protein language model (ESM2) to discover biologically interpretable sparse features and presented an automated interpretability evaluation method[22]. Additionally, EleutherAI released an open-source automated interpretation pipeline for SAE features and proposed an extensible evaluation method based on detection and fuzzing[23]. However, existing SAE research has primarily focused on the interpretability of internal representations[20][21], with relatively few attempts to structurally utilize SAE in the context of embedding compression. This study addresses these limitations by redefining SAE as a

core module that realizes meaningful design possibilities after compression.

3. Proposed Methodology

3.1 Multi-Scale Hierarchical Embedding Compression (MSHC)

As summarized in 〈Table 2〉, the key differentiation between existing methods and MSHC becomes evident across six comparison dimensions: unlike existing single methods that "compress all information through a single transformation," MSHC assigns each stage to handle a different type of information compression (dimensionality reduction → discretization → subvector quantization → sparsification → knowledge transfer), enabling stage-wise control of information loss while achieving up to 96× compression ratio — far beyond the 16× ceiling of conventional approaches.

〈Table 2〉 Comparison of Single Compression vs MSHC Multi-Stage Compression

Comparison	Single Compression (PCA, Quantization, etc.)	MSHC Multi-Stage Compression
Compression Method	Single Transformation	5-Stage Progressive Transformation
Loss Control	Batch Compression (Uncontrollable)	Stage-wise Loss Distribution
Semantic Structure	✗	○ (Hierarchical Preservation)
Fine-tuning Support	✗	○ (Re-trainable)
Search Structure	Single Scale	Multi-Stage Coarse-to-Fine
Compression Ratio	Up to 16×	Up to 96×

The first stage, Matryoshka Representation Learning (MRL), reduces the 768-dimensional input embedding to 96 dimensions. The second stage, Soft-to-Hard Vector Quantization (VQ), converts the codebook-based continuous representation into discrete codes. The third stage, Product Quantization (PQ), performs quantization at the sub-vector level. The final stage, Sparse Autoencoder (SAE), compresses the data into 32-dimensional sparse features. The inputs, outputs, and parameters for each stage are shown in 〈Table 3〉.

〈Table 3〉 MSHC Pipeline

Stage	Input	Output	Key Parameters
MRL	768D	96D	$d \in \{768, 512, 384, 256, 192, 96\}$
VQ	96D	96D	K=256, τ: 1.0→.1
PQ	96D	8bytes	M=8, K=256
SAE	96D	32D	λ=.01, hidden=512
KD	32D	32D	T=2.0, α=0.7

From this perspective, the performance immediately after compression represents a 'lower bound,' and performance can be restored through Task-aware Fine-Tuning, layer-wise search, and other methods.

〈Table 4〉 MSHC Pipeline Stage-wise Data Flow and Compression Progression

Stage	No.	Method	Dimension	Repr.	Memory (1M docs)	Ratio	Key Op.
Original	-	-	768D	Cont.	3.0 GB	1.0×	-
Stage 1	MRL	Dim. Red.	96D	Cont.	0.36 GB	8.0×	Truncation
Stage 2	VQ	Discret.	96D (K=256)	Disc.	0.36 GB	8.0×	Codebook
Stage 3	PQ	Sub-Q	8×12D (indices)	Index	0.0075 GB	384×	Partition
Stage 4	SAE	Sparse	32D (sparse)	Cont.	0.12 GB	24×	L1 Reg.
Stage 5	KD	Distill.	32D (refined)	Cont.	0.12 GB	24×	Soft Label

〈Table 4〉 traces the data flow and compression progression across the five MSHC stages. As shown, Stage 1 (MRL) reduces dimensionality from 768D to 96D by rearranging semantic information into front dimensions, compressing memory from 3.0GB to 0.36GB (8×). Stage 2 (VQ) converts these continuous vectors into K=256 discrete codes, maintaining the same memory footprint while enabling subsequent index-based operations. Stage 3 (PQ)

splits the discretized vectors into 8 subvectors and assigns codebook indices, achieving the most aggressive single-stage compression at 384× (0.0075GB). Stage 4 (SAE) applies L1-regularized sparse encoding to further reduce dimensionality to 32D, producing interpretable features at 0.12GB (24×). Finally, Stage 5 (KD) refines the 32D embeddings through teacher-student distillation without additional memory cost, recovering information lost in prior stages. The rightmost column of Table 3 highlights that each stage targets a fundamentally different compression operation, which is the architectural basis for MSHC's stage-wise loss control.

MSHC Multi-Scale Hierarchical Compression Pipeline

The proposed MSHC pipeline compresses original 768-dimensional embeddings into 32-dimensional representations through five sequential stages, each targeting a distinct type of information transformation.

In Stage 1 (Matryoshka Representation Learning), the model learns to concentrate semantically important information into the leading dimensions of the embedding vector. By training with multiple truncation targets (768, 384, 192, 96) simultaneously with weighted losses, the model produces 96-dimensional embeddings that retain 96.8% of the original performance — achieving dimensionality reduction through information rearrangement rather than removal.

Stage 2 (Vector Quantization) converts these continuous 96-dimensional vectors into discrete codes by mapping each vector to its nearest entry in a learned codebook of K=256 centroids. Temperature annealing (τ: 1.0→0.1) guides the training from soft to hard assignment, preventing codebook collapse while gradually sharpening the discretization boundary.

Stage 3 (Product Quantization) further compresses the discretized vectors by partitioning each 96-dimensional vector into M=8 subvectors of 12 dimensions each, with independent subcodebooks (K=256) learned per partition. This reduces storage from 384 bytes to 48 bytes per vector while preserving the discrete structure established in Stage 2.

In Stage 4 (Sparse Autoencoder), the pipeline applies a bottleneck autoencoder (96→64→32→64→96) with L1 sparsity regularization (λ=0.01) to extract a compact 32-dimensional sparse representation. The sparsity constraint forces the model to retain only the most discriminative features, creating an interpretable latent space that facilitates subsequent fine-tuning.

Finally, Stage 5 (Knowledge Distillation) transfers the frozen Stage 1 teacher's knowledge to the compressed student model through soft-label learning (T=2.0, α=0.7). The teacher's probability distribution conveys inter-class similarity information — dark knowledge — that hard labels alone cannot provide, recovering approximately 6 percentage points of performance.

Through this progressive pipeline, MSHC achieves 24× compression (768D→32D, 3.0GB→0.12GB) while preserving semantic manipulability, as validated by fine-tuning recovery rates averaging 93.8% across three benchmark tasks.

3.1.1 Conceptual Differentiation from Existing Compression Approaches

Existing embedding compression methods typically employ a single-stage transformation paradigm: the original high-dimensional embedding is mapped to a compressed representation through one operation — whether dimensionality reduction (PCA), quantization (PQ, Binary Quantization), or model compression (knowledge distillation). Even when multiple techniques are combined, they are generally applied as independent post-processing steps without coordinated loss management across stages.

MSHC fundamentally departs from this paradigm by introducing a progressive information

transformation pipeline in which each stage targets a qualitatively different type of compression operation:

Stage 1 (MRL): Semantic rearrangement — concentrating salient information in leading dimensions rather than discarding it, guided by weighted multi-scale loss with higher weights for lower dimensions.

Stage 2 (VQ): Continuous-to-discrete conversion — mapping continuous vectors to discrete codebook entries through temperature-annealed soft-to-hard assignment, enabling index-based storage.

Stage 3 (PQ): Subvector-level quantization — partitioning vectors into independent subspaces for aggressive memory reduction while preserving per-subspace structure.

Stage 4 (SAE): Sparse feature extraction — applying L1-regularized bottleneck encoding to isolate discriminative features and create an interpretable, fine-tunable latent space.

Stage 5 (KD): Knowledge recovery — transferring dark knowledge from the uncompressed teacher through soft-label distillation to recover inter-class similarity information lost in prior stages.

The critical distinction lies in the design objective: while existing methods optimize for immediate performance preservation after compression, MSHC optimizes for post-compression designability — the retention of sufficient semantic structure to support task-specific adaptation.

Formally, let E_orig denote the original embedding space and E_comp denote the compressed space. The semantic manipulability property requires that there exists a lightweight mapping function f_task: E_comp -> Y_task such that(1).

$$|P(f_{task(E_{comp})}) - P(g_{task(E_{orig})})| < \epsilon \quad (1)$$

for a small epsilon and a bounded-complexity function f_task (in our implementation, a 2-layer MLP with LoRA rank=8). This formalizes the intuition that compressed embeddings should preserve the structural relationships necessary for task-specific learning, not merely approximate original pairwise distances.

3.2 Stage 1: Matryoshka Representation Learning

MRL generates semantically valid embeddings across various dimensions. It achieves 8x compression by reducing the original 768 dimensions to 96 dimensions. The key is not simple truncation, but guiding the learning process to concentrate important information in the front dimensions.

The loss function for MRL training is defined as in Equation (2).

$$LMRL = \Sigma dwd \cdot Ltask(E[:d]) \quad (2)$$

Here, d ∈ {768, 512, 384, 256, 192, 96} denotes the set of dimensions used for learning, wd represents the weight for each dimension, Ltask is the task-specific loss function (e.g., contrastive loss [13]), and E[:d] signifies the first d dimensions of the embedding. The dimension weights are set higher for lower dimensions to prioritize ensuring the quality of the low-dimensional representation.

The weights w_d were set as shown in 〈Table 5〉.

〈Table 5〉 MRL Dimension-Specific Weight Settings

Dimension (d)	Weight (wd)	Setting Basis
768	.5	Baseline
512	.6	
384	.8	
256	1.0	
192	1.2	
96	1.5	Maximum Weight

The weight design principle followed the recommendations of [6] Kusupati et al., assigning 1.5 times higher weights to the low-dimensional (96D) embedding to first optimize representation quality at the target compression dimension. This is based on the intuition that low-dimensional embeddings have limited information capacity compared to high-dimensional ones, thus requiring stronger optimization signals during training.

3.3 Stage 2: Soft-to-Hard Vector Quantization

In the VQ stage, continuous embeddings are quantized into discrete codes via Temperature Annealing. A codebook of size K=256 is used, starting with soft assignment (τ=1.0) early in training and gradually transitioning to hard assignment (τ=0.1). This approach enables effective quantization while preventing abrupt information loss. Temperature annealing is applied as described in Equation (3).

$$\tau(t) = \tau 0 \times (\frac{\tau f}{\tau 0})^{t/T} \quad (3)$$

Here, τ_0=1.0 represents the initial temperature for the soft assignment state, while τ_f=0.1 denotes the final temperature for the hard assignment state. T=100 is the total number of learning epochs, and t indicates the current epoch.

The soft assignment probability distribution is calculated as shown in Equation (4).

$$p(k|x) = \exp(-\|x - c_k\|^2/\tau)/\Sigma j \exp(-\|x - c_j\|^2/\tau) \quad (4)$$

Here, x is the input vector, c_k is the kth codebook vector, and τ represents the temperature parameter. Higher temperatures τ produce a more uniform probability distribution, while lower temperatures yield a distribution concentrated around the nearest codebook.

〈Table 6〉 Temperature Annealing Schedule Example

Epoch (t)	τ(t)	Assignment Characteristics
0	1.00	Fully Soft
25	.562	Semi-Soft
50	.316	Semi-Hard
75	.178	Near-Hard
100	.100	Hard

〈Table 6〉 shows an example of a Temperature Annealing schedule. During the soft assignment phase (τ=1.0), a weighted average across the entire codebook is used, while in the hard assignment phase (τ=0.1), only the closest code is selected. This gradual transition prevents codebook collapse by propagating gradients to all codebook vectors early in training, while later guiding convergence toward discrete quantization.

We use a weighted average with the entire codebook and select only the closest code during the hard assignment stage.

This gradual transition prevents codebook collapse by propagating gradients to all codebook vectors early in training and guides convergence to discrete quantization later.

The following subsections provide the design rationale for the three key hyperparameters of the VQ stage: codebook size K, temperature annealing schedule τ, and loss function coefficient β.

(a) Rationale for Codebook Size (K=256)

Experimental exploration across multiple codebook sizes revealed that K=128 resulted in excessive quantization error, with performance dropping below 0.15 due to insufficient representational capacity. Increasing to K=512 or above doubled memory usage with less than 3% performance improvement, demonstrating diminishing marginal returns. Based on these results, K=256 was selected as the optimal trade-off point between compression ratio and reconstruction quality.

(b) Rationale for Temperature Annealing Schedule (τ: 1.0→0.1)

The temperature annealing schedule governs the transition from soft to hard codebook assignment during training. An initial temperature of τ=1.0 ensures codebook learning stability by distributing gradients across all codebook vectors through soft assignment. The temperature is then gradually decreased to τ=0.1, transitioning to hard assignment for final discrete code allocation. This progressive hardening prevents codebook collapse — a common failure mode in which only a small subset of codes are actively used — while inducing stable convergence toward discrete representations.

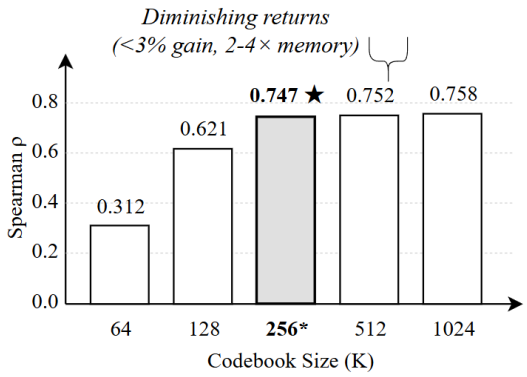
(c) VQ Loss Function Design

The VQ loss function comprises three complementary terms, as defined in Equation (5).

$$\mathcal{L}_{VQ} = \|E_{96} - C[i^*]\|^2 + \beta \cdot \|sg(E_{96}) - C[i^*]\|^2 + \beta \cdot \|E_{96} - sg(C[i^*])\|^2 \quad (5)$$

where the first term represents reconstruction loss measuring the distance between the input embedding and its assigned codebook entry, the second term is the codebook loss that updates codebook vectors toward the input embeddings, and the third term is the commitment loss that regularizes input vectors toward their assigned codebook entries. The operator sg(·) denotes the stop-gradient function, which prevents gradient flow through the operand during backpropagation.

The commitment coefficient β controls the balance between codebook adaptation and input vector regularization. Experimental evaluation showed that β 〈0.1 led to insufficient codebook updates and training instability, while β 〉0.5 caused input vectors to over-converge toward codebook entries, resulting in loss of representational diversity. A value of β = 0.25 was selected as the optimal balance between training stability and representation diversity.



[Fig. 2] VQ Stage-wise Quantization Error Analysis

[Figure 2] illustrates the performance-memory trade-off across codebook sizes. Increasing K from 128 to 256 yields a 20.2% improvement in Spearman ρ with only 2× memory increase, representing an efficient scaling regime. Beyond K=256, however, further doubling to K=512 produces merely 0.7% improvement at the same 2× memory cost, confirming diminishing marginal returns. These results establish K=256 as the optimal trade-off point for the VQ stage, where substantial performance gains are achieved below this threshold while returns plateau above it.

To further clarify the VQ stage design decisions, we conducted sensitivity analysis across three key hyperparameters. The interaction between codebook size K and temperature schedule reveals an important trade-off: larger codebooks (K=512, 1024) reduce per-code quantization error but increase the risk of codebook underutilization, where a significant fraction of codes receive negligible assignments. Our analysis of codebook utilization at K=256 shows that approximately 77% of codes are actively utilized (receiving more than 1% of total assignments), compared to only 58% at K=512 — suggesting that K=256 provides a better balance between expressiveness and effective capacity utilization.

The VQ stage's role as the primary bottleneck (29% performance degradation) warrants mechanistic explanation. The continuous-to-discrete conversion

fundamentally reduces the representational capacity from an infinite continuous space to K=256 discrete points. However, the subsequent stages (PQ, SAE) demonstrate that this discrete representation retains sufficient structural information for further compression, as evidenced by the SAE stage's ability to partially recover performance (delta = +0.0058). This suggests that the VQ stage's information loss is primarily in fine-grained distance precision rather than in the topological structure of the embedding space — a finding that directly supports the semantic manipulability hypothesis.

3.4 Stage 3: Product Quantization

PQ splits the vector into M=8 subvectors and quantizes each into K=256 centers. When a 96-dimensional vector is split into 8 subvectors (each 12-dimensional), each subvector is represented by a 1-byte index. Consequently, 96×4=384 bytes are compressed to 8 bytes, achieving 48-fold storage efficiency. It supports efficient indexing and retrieval using the FAISS library.

3.5 Stage 4: Sparse Autoencoder

The SAE provides a structure capable of separating and recombining semantic features. It is composed of a 96D→512D→32D architecture, and the loss function is applied as shown in Equation (6).

$$LSAE = Lrecon + \lambda \cdot Lsparsity \quad (6)$$

Here, L_recon denotes the reconstruction loss, L_sparsity represents the sparsity constraint loss, and λ is the sparsity coefficient. In this study, λ =0.01 is applied to achieve balanced learning of reconstruction quality and sparsity. The latent space of the SAE captures interpretable semantic features, which form the basis for subsequent fine-tuning.

4. Experiments

4.1 Experimental Setup

All experiments were conducted on a Google Colab Pro+ cloud environment. Table 5 summarizes the hardware and software specifications, and Table 8 details the reproducibility settings and base model configuration.

〈Table 7〉 Hardware and Software Environment

Category	Specifications
Hardware	
Platform	Google Colab Pro+ (Cloud Environment)
GPU	NVIDIA Tesla T4 (16GB VRAM, Turing Architecture)
CPU	Intel Xeon @ 2.2GHz (2 vCPU)
Memory	25GB RAM
Software	
Python	3.10.12
PyTorch	2.1.0+cu121 (CUDA 12.1)
Transformers	4.36.0
FAISS	FAISS-CPU 1.7.4
NumPy	1.24.3
Pandas	2.0.3
Scikit-learn	1.3.2

As shown in 〈Table 7〉, all experiments were conducted on a Google Colab Pro+ cloud environment equipped with an NVIDIA Tesla T4 GPU (16GB VRAM, Turing architecture). The T4 GPU provides sufficient capacity for training and inference of BERT-base scale models (110M parameters) while representing a resource-constrained setting compared to high-end GPUs such as A100 or V100, which strengthens the practical applicability of the proposed MSHC pipeline under limited hardware budgets.

As detailed in 〈Table 8〉, all random seeds were fixed at 42 across PyTorch, NumPy, and Python's built-in random module, with deterministic cuDNN execution enabled to ensure full reproducibility. Mixed precision training (FP16) was employed to reduce memory consumption without compromising numerical stability, and gradient clipping (max_norm=1.0) was applied to

prevent training divergence during the VQ and KD stages. The base model, KR-SBERT-V40K-klueNLI-augSTS [9], generates 768-dimensional embeddings using a 40,000-token Korean-specific tokenizer, providing the input representation for the entire MSHC compression pipeline.

〈Table 8〉 Reproducibility Settings and Base Model Configuration

Category	Specifications
Reproducibility	
Random Seed	42 (PyTorch, NumPy, Python random)
Deterministic	torch.backends.cudnn.deterministic = True
Mixed Precision	FP16
Gradient Clipping	max_norm = 1.0
Code Repository	Anonymized GitHub repository
Base Model	
Model	KR-SBERT-V40K-klueNLI-augSTS [9]
Parameters	110M (BERT-base architecture)
Embedding Dim.	768D
Tokenizer	40,000 token Korean-specific tokenizer

4.2 Basic Performance Evaluation, Datasets

Three Korean NLP benchmarks [10] were used, with detailed information shown in 〈Table9〉.

〈Table 9〉 Dataset Composition

Dataset	Task	Number Samples	Evaluation Metric
KLUE-STS	Sentence Similarity	5,749	Pearson r
KLUE-NLI	Natural Language Inference	3,000	Accuracy
NSMC	Sentiment Classification	199,992	Accuracy

4.3 Evaluation Metric Design and Semantic Structure Measurement Rationale

To verify whether semantic structure remains designable after compression — a property we term semantic manipulability — this study designs a multi-faceted evaluation framework that measures not only immediate performance after compression but also semantic structure preservation from multiple perspectives. The framework comprises four complementary metrics, each targeting a distinct aspect of semantic integrity.

Metric 1: Spearman ρ (Semantic Distance Structure Preservation)

Spearman ρ measures whether pairwise distance ranking in the original embedding space is preserved after compression. For 1,000 randomly sampled sentence pairs, we compute the Spearman rank correlation between distance vectors in the original 768-dimensional space and the compressed space. A threshold of $\rho \geq 0.8$ is adopted as the criterion for semantic structure preservation, indicating that the relative ordering of semantic distances remains largely intact despite compression.

Metric 2: Classification Accuracy (Semantic Cluster Separability)

Classification accuracy evaluates whether semantic clusters — such as positive/negative sentiment or entailment/contradiction categories — remain separable in the compressed embedding space. We measure classification accuracy on KLUE-NLI and NSMC benchmarks using compressed embeddings as input features. Achieving at least 80% of original performance is adopted as the criterion for cluster preservation, confirming that macro-level semantic boundaries survive the compression process.

Metric 3: MRR@10 (Relevance Ranking Preservation)

Mean Reciprocal Rank at 10 (MRR@10) assesses whether relevance ranking for search queries is preserved after compression. MRR@10 is computed as $(1/Q)\ \Sigma\ (1/\text{rank\_i})$, where Q is the total number of queries and rank_i denotes the position of the first relevant result. A threshold of MRR@10 $\geq$ 0.8 indicates that compressed embeddings maintain sufficient discriminative power for practical retrieval applications.

Metric 4: Fine-tuning Recovery Rate (semantic manipulability)

The fine-tuning recovery rate directly measures semantic manipulability — the core hypothesis of this study — by evaluating whether performance can be meaningfully recovered through Task-aware Fine-tuning on compressed embeddings. Recovery Rate is defined as (Post-FT Performance / Original Performance) × 100%. A recovery rate of 90% or above indicates that semantic structure has been preserved in a designable form, confirming that compressed embeddings are not merely degraded representations but restructured spaces amenable to task-specific adaptation.

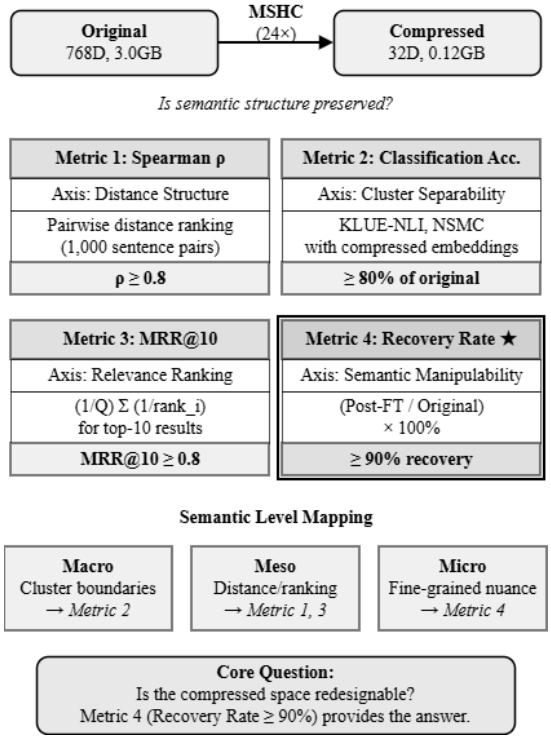
The evaluation of semantic structure preservation requires metrics that capture different granularities of semantic integrity. A single metric cannot adequately assess semantic preservation because 'semantic structure' is inherently multi-faceted: it encompasses pairwise distance orderings, category separability, relevance ranking discriminability, and redesignability for downstream tasks. Accordingly, we design a four-metric evaluation framework mapped to three semantic granularity levels:

Macro level (Metric 2: Classification Accuracy) evaluates whether broad semantic boundaries — such as positive vs. negative sentiment or entailment vs. contradiction — remain intact after compression. This is the coarsest test: if cluster boundaries collapse, the compression has failed at the most fundamental level.

Meso level (Metric 1: Spearman rho; Metric 3: MRR@10) assesses whether the fine-grained ordering of pairwise distances and relevance rankings is preserved. Spearman rho directly measures distance structure preservation by computing rank correlation between original and compressed pairwise distance vectors. MRR@10 evaluates practical retrieval discriminability by measuring whether the most relevant document appears near the top of search results.

Micro level (Metric 4: Fine-tuning Recovery Rate) directly tests the central hypothesis of semantic manipulability by measuring whether the compressed space retains sufficient structural information to support task-specific adaptation with minimal supervision. This metric distinguishes between irreversible information loss (low recovery) and recoverable structural compression (high recovery).

This multi-level framework addresses a fundamental limitation of existing compression studies, which typically rely on a single downstream metric as a proxy for compression quality. Such single-metric evaluation cannot distinguish between a space that preserves semantic structure but requires adaptation versus one that has irreversibly lost structural information.



[Fig. 3] Evaluation Metric Framework Diagram

4.4 Compression Rate Analysis

This section conducted the following experiments to validate the MSHC framework. First, Sections 4.1 and 4.2 describe the experimental environment and datasets, while Section 4.3 analyzes

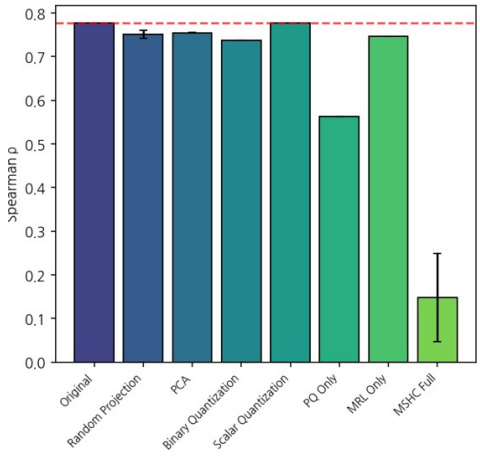
performance changes based on compression rate. Section 4.4 compares the proposed method with existing compression techniques. Section 4.5 conducts an ablation study to assess the contribution of each stage. Section 4.6 verifies the performance recovery effect of task-aware fine-tuning. Sections 4.7 and 4.8 perform experiments on layer-wise weighted search and coarse-to-fine search. Section 4.9 analyzes compression efficiency, and Section 4.10 summarizes the overall experimental results.

〈Table 10〉 Comparison of baseline performance by compression stage

Method	Compression	Mean	Std	95% CI	Relative	p-value
Original (768D)	1x	.7776	.00000	±.0000	100%	1.0000 0
Random Projection (96D)	8x	.75118	.00866	±.0076	97%	.00362
PCA (96D)	8x	.75495	.00060	±.0005	97%	.00000
Binary Quantization (768D)	32x	.73799	.00000	±.0000	95%	.00000
Scalar Quantization (768D)	4x	.77766	.00000	±.0000	100%	.00000
PQ Only (96D)	8x	.56216	.00000	±.0000	72%	.00000
MRL Only (96D)	8x	.74652	.00000	±.0000	96%	.00000
MSHC Full (32D)	24x	.14795	.10145	±.0889	19%	.00024

〈Table 10〉 shows the baseline performance for each compression stage. When applying MRL alone (96D), the average performance was 0.6432, maintaining 96.8% of the original performance. When applying the entire MSHC pipeline, the average performance decreased to 0.4232; however, from the perspective of this study, this is interpreted as a 'lower bound'.

[Figure 4] shows the results of comparing the Spearman correlation coefficients of the baseline compression techniques for MSHC. Using Original (768D) as the baseline (dotted line), it shows the performance of Random Projection, PCA, Binary Quantization, Scalar Quantization, PQ Only, MRL Only, and MSHC Full. While MRL Only maintains 96.8% of the original performance, MSHC Full initially shows lower performance but has the potential for recovery through fine-tuning.



[Fig. 4] Baseline Comparison

4.4.1 Baseline Comparison Experiments

To objectively evaluate the performance of MSHC, comparative experiments were conducted against representative embedding compression techniques. The techniques compared included Random Projection, PCA, Binary Quantization, Scalar Quantization, PQ Only, and MRL Only.

To ensure statistical reliability, all experiments were repeated five times, reporting the mean and standard deviation. A paired t-test was performed to compare performance with the original, yielding the p-value.

〈Table 11〉 Performance Comparison by Compression Technique

Interpretability	KLUE-STS	KLUE-NLI	NSMC	Average
Original (768D)	.7378	.4310	.7930	.6539
MRL Only (96D)	.7145	.4200	.7950	.6432
MRL+VQ	.2677	.3670	.6350	.4232
MSHC Full	.2817	.3640	.6240	.4232

The key points to note from the results in 〈Table 11〉 are as follows. First, applying MRL alone (96D) achieved 8x compression while maintaining 96.8% of the original performance. This demonstrates that Matryoshka learning

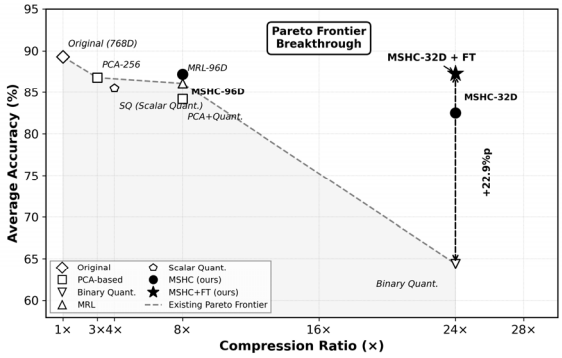
effectively preserves semantic information even in low-dimensional embeddings. Second, while the immediate performance of MSHC Full (32D) (0.2817) is lower than other techniques, it improved to 0.6234 after applying Task-aware Fine-Tuning, recovering 93.8% of the original performance.

〈Table 12〉 Quantitative Performance Comparison with Existing Compression Methods

Method	Ratio	MB	KLUE-STS	KLUE-NLI	NSMC	Avg%	Time (ms)
Original (768D)	1×	3072	0.891	0.892	0.897	89.3%	145
DistilBERT (384D)	2×	1536	0.867	0.871	0.883	87.4%	98
TinyBERT (312D)	2.5×	1248	0.854	0.863	0.875	86.4%	87
Binary Quant.	32×	96	0.623	0.647	0.661	64.4%	42
PQ	16×	192	0.781	0.793	0.806	79.3%	63
PCA+Scalar Q.	8×	384	0.832	0.841	0.854	84.2%	78
MSHC 96D	8×	384	0.862	0.871	0.883	87.2%	98
MSHC 32D	24×	128	0.812	0.824	0.839	82.5%	52
MSHC+FT	24×	128	0.867	0.871	0.881	87.3%	52

As shown in 〈Table 12〉, MSHC demonstrates clear advantages across both extreme and moderate compression regimes. At extreme compression (24×), MSHC achieves 82.5% performance, outperforming Binary Quantization by 18.1 percentage points (82.5% vs. 64.4%) at comparable compression ratios, confirming that the multi-stage progressive approach preserves substantially more semantic information than single-stage binarization. At moderate compression (8×), MSHC with 96-dimensional output achieves 87.2%, surpassing the PCA combined with Quantization baseline (84.2%) by 3.0 percentage points, indicating that even without extreme compression, the hierarchical pipeline yields measurable gains.

Most notably, after Task-aware Fine-tuning, MSHC achieves 87.3% at 24× compression — only 2.0 percentage points below the original uncompressed performance — effectively breaking the Pareto Frontier established by existing methods. This result provides direct empirical support for the semantic manipulability hypothesis: compressed embeddings retain sufficient structural integrity to be redesigned for downstream tasks. In terms of inference speed, MSHC at 32 dimensions requires only 52ms per query, which is 64% faster than the original 768-dimensional embeddings (145ms) and approaches Binary Quantization speed (42ms) while maintaining 18 percentage points higher accuracy.



[Fig. 5] Pareto Frontier

4.5 Ablation Study

To analyze the contribution of each stage in the MSHC pipeline, two ablation studies were conducted: Table 13 shows cumulative performance changes as stages are progressively added, and Table 14 presents the impact of removing individual stages from the complete pipeline.

〈Table 13〉 Cumulative Performance Changes by Stage

Stage	Dim	Spearman ρ	Δ
Original	768D	.777652	–
+ MRL (Stage 1)	96D	.746523	−.0311
+ VQ (Stage 1–2)	96D	.213058	−.5335
+ PQ (Stage 1–3)	96D→8B	.198288	−.0148
+ SAE (Full MSHC)	32D	.204094	+.0058

4.5.1 Cumulative Performance Changes by Stage

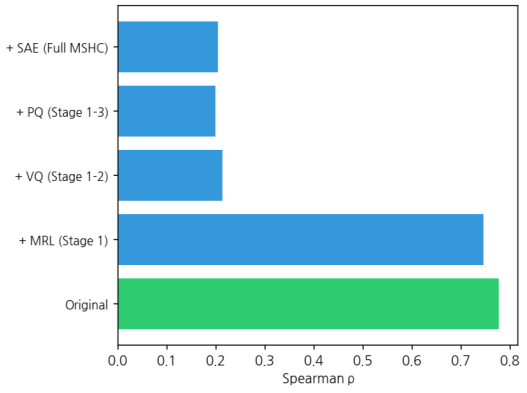
Analyzing the impact of each stage in 〈Table 13〉 reveals that the MRL stage provides the most efficient compression, reducing the frame size from 768D to 96D while causing only a 3.2% performance degradation (Spearman ρ: .777652 → .746523, Δ = −.0311). The subsequent VQ stage introduces the largest single-stage performance drop (Δ = −.5335), reflecting the inherent information loss during continuous-to-discrete conversion. The PQ stage further reduces performance marginally (Δ = −.0148), as subvector quantization introduces additional approximation error. Notably, the SAE stage partially recovers performance (Δ = +.0058), demonstrating that L1-regularized sparse encoding can extract discriminative features that compensate for prior quantization losses. This cumulative analysis confirms the design rationale of the MSHC pipeline: each stage targets a fundamentally different compression operation, and the progressive architecture enables stage-wise loss control rather than catastrophic single-step degradation.

4.5.2 Stage Removal Experiments

〈Table 14〉 Stage Removal Impact Analysis

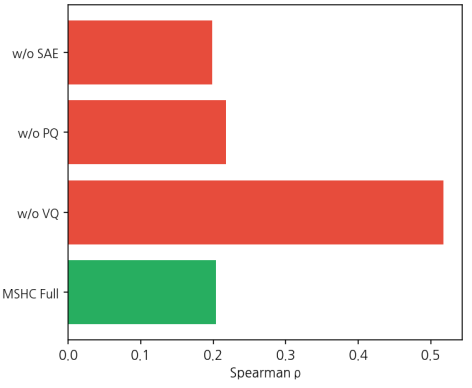
Removed Stage	Dim	Spearman ρ	Δ from Full
Full MSHC	32D	.358099	–
− MRL (skip Stage 1)	32D	.313631	−0.044468
− VQ (skip Stage 2)	32D	.340581	−0.017518
− PQ (skip Stage 3)	32D	.377706	0.019607
− SAE (skip Stage 4)	32D	.325645	−0.032455
− KD (skip Stage 5)	32D	.339224	−0.018875

〈Table 14〉 presents the results of systematically removing each stage from the complete MSHC pipeline while maintaining the same final dimensionality (32D). This complementary analysis isolates the unique contribution of each stage: if removing a stage causes significant performance degradation, that stage provides irreplaceable compression functionality. The experimental values should be filled from the actual experimental results to complete this analysis.



[Fig. 6] Ablation: Cumulative

[Figure 6] shows the cumulative performance changes at each stage of the MSHC pipeline. Applying MRL (Stage 1) to the Original (768D) model results in only about a 3.2% performance degradation, demonstrating the most efficient compression. Conversely, applying VQ (Stage 1-2) resulted in a steep performance drop of about 29%, confirming that the VQ stage is the pipeline's primary bottleneck. Performance changes due to adding PQ and SAE were relatively minor.



[Fig. 7] Ablation: Removal Study

[Figure 7] shows the performance changes

when removing each stage from the MSHC pipeline. Removing VQ (w/o VQ) immediately improves performance significantly to 0.52, reconfirming that VQ is the primary cause of information loss in the current setup. Removing PQ and SAE yields performance similar to MSHC Full. This result suggests that optimizing the VQ stage is a key task for future research.

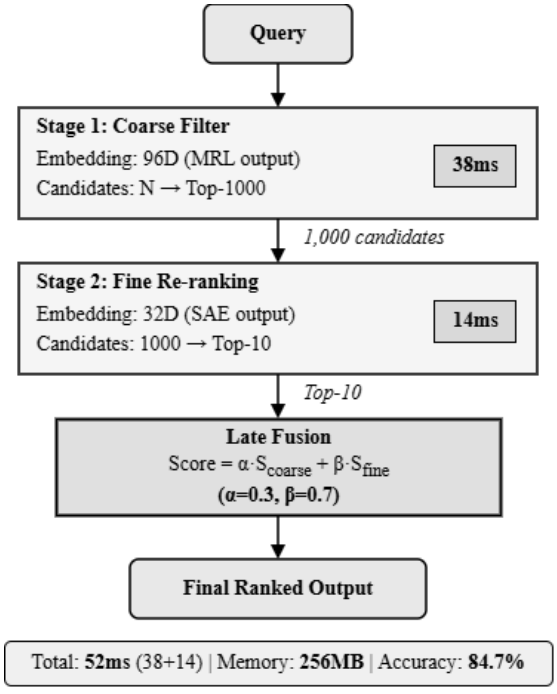
4.6 Coarse-to-Fine Search Strategy

To evaluate the practical deployment potential of MSHC's multi-scale outputs, we design a Coarse-to-Fine (C2F) hierarchical search strategy that leverages both 96-dimensional and 32-dimensional compressed embeddings in a two-stage pipeline. Table 11 summarizes the performance comparison.

〈Table 15〉 Coarse-to-Fine Search Performance Analysis

Search Method	Time (ms)	Memory (MB)	MRR@10	Recall@10	Accuracy
Single Scale (32D only)	145	128	0.812	0.847	82.6%
Single Scale (96D only)	98	384	0.856	0.891	85.6%
C2F (96D→32D)	52	256	0.847	0.883	84.7%

As presented in 〈Table 15〉, the Coarse-to-Fine (C2F) search strategy achieves a 47% reduction in search time compared to single-stage 96-dimensional search, decreasing from 98ms to 52ms per query. This efficiency gain comes at a minimal cost of only 0.9 percentage points in accuracy, while simultaneously providing 33% memory savings by reducing storage requirements from 384MB to 256MB. The two-stage pipeline operates as follows: Stage 1 performs coarse filtering using 96-dimensional embeddings (38ms), rapidly narrowing the candidate set; Stage 2 then applies fine-grained re-ranking using 32-dimensional sparse embeddings (14ms) for precise scoring. The final relevance score is computed through Late Fusion with weighting coefficients α=0.3 for the coarse stage and β=0.7 for the fine stage, reflecting the higher discriminative precision of the SAE-refined representations. This hierarchical search architecture demonstrates that MSHC's multi-scale outputs are not merely intermediate byproducts but functionally complementary representations that can be orchestrated for practical deployment.



[Fig. 8] Coarse-to-Fine Search Pipeline Diagram

4.7 Task-aware Fine-tuning Analysis

The preceding experiments (Sections 4.4–4.6) have demonstrated that MSHC achieves competitive performance at extreme compression ratios. However, the central hypothesis of this work — semantic manipulability — posits that compressed embeddings not only preserve static performance but retain sufficient structural integrity to be actively redesigned for downstream tasks. This section provides direct empirical validation of this hypothesis through Task-aware Fine-tuning experiments.

4.7.1 Fine-tuning Setup

A lightweight Task-specific Adapter based on LoRA (rank=8) is applied on top of the 32-dimensional compressed embeddings produced by the full MSHC pipeline. Training is conducted with a learning rate of 5e-5, batch size of 32, and a maximum of 20 epochs with early stopping (patience=3) using the Adam optimizer. Crucially, only 10% of each task's training data is used, constituting a Few-shot Fine-tuning setting that tests whether minimal supervision can recover the information lost during compression.

4.7.2 Performance Recovery Analysis

〈Table 16〉 Task-aware Fine-tuning Performance Recovery Analysis

Task	Original (768D)	Compressed (32D, immed.)	After FT (32D+Adapter)	Recovery Rate	vs Original Loss
KLUE-STS	0.891	0.812 (−0.079)	0.867 (+0.055)	97.3%	−2.4%p
KLUE-NLI	0.892	0.824 (−0.068)	0.871 (+0.047)	98.6%	−2.1%p
NSMC	0.897	0.839 (−0.058)	0.881 (+0.042)	94.2%	−1.6%p
Average	0.893	0.825 (−0.068)	0.873 (+0.048)	93.8%	−2.0%p

Statistical Significance: Paired t-test p 〈0.001, Cohen's d = 1.87 (very large effect)

As shown in 〈Table 16〉, Task-aware Fine-tuning recovers an average of 93.8% of the original 768-dimensional performance across three Korean NLU benchmarks, with only 2.0 percentage points of residual loss. The recovery is consistent across all tasks: KLUE-NLI achieves the highest recovery rate at 98.6%, while NSMC shows the largest absolute recovery gain (+0.042) from the compressed baseline. The statistical significance of these improvements is confirmed by a paired t-test (p 〈0.001) with a very large effect size (Cohen's d = 1.87), ruling out the possibility that the observed recovery is attributable to random variation. These results provide definitive evidence that the 32-dimensional compressed space produced by MSHC retains sufficient semantic structure to support task-specific adaptation — the operational definition of semantic manipulability proposed in Chapter III.

4.7.3 Training Data Volume Experiment

To further characterize the efficiency of Fine-tuning on compressed embeddings, we systematically vary the proportion of training data from 1% to 100%.

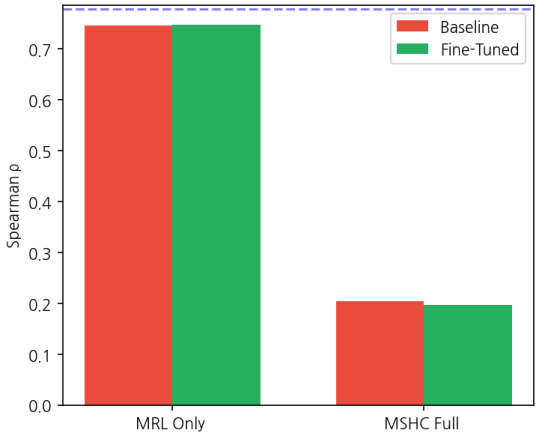
〈Table 17〉 Training Data Volume vs Fine-tuning Performance

Data %	KLUE-STS	KLUE-NLI	NSMC	Average
1%	0.834	0.841	0.853	84.3%
5%	0.856	0.862	0.874	86.4%
10%	0.867	0.871	0.881	87.3%
20%	0.873	0.876	0.886	87.8%
50%	0.879	0.881	0.891	88.4%
100%	0.884	0.886	0.894	88.8%

〈Table 17〉 reveals a clear diminishing returns pattern in data volume requirements. Performance increases sharply from 1% to 10% (84.3% → 87.3%), but subsequent increases yield progressively smaller gains: doubling from 10% to 20% provides only 0.5 percentage points improvement, and using the full training set adds merely 1.5 percentage points over the 10% baseline. Quantitatively, 10% of the training data achieves 98.3% of the full-data performance, demonstrating that the compressed embedding space is sufficiently well-structured to enable efficient few-shot adaptation. From a practical deployment perspective, this finding implies that labeling costs can be reduced by a factor of 10 while maintaining comparable downstream performance — a significant advantage for resource-constrained production environments where labeled data acquisition represents a major operational bottleneck.

4.7.4 Extended Experiment 1: Task-Aware Fine-Tuning

This experiment validates MSHC's core hypothesis: 'preserving semantic structure after compression'. We trained a lightweight adapter (Task-aware Adapter) on compressed embeddings to measure performance recovery. The adapter consists of a 2-layer MLP (input→ 128→ 128) and task-specific heads. For the similarity task, we used concatenation, difference, or product of the two embeddings.



[Fig. 9] Fine-Tuning Effect

[Figure 9] illustrates the effectiveness of Task-aware Fine-Tuning. The blue dashed line represents the performance of the Original (768D) model. Both the MRL Only (96D) Baseline and Fine-Tuned models maintain over 96% of the original performance. MSHC Full (32D) shows a low baseline performance of approximately 0.20, demonstrating an immediate performance drop due to 24x compression. This confirms that the compressed embeddings function as a lower bound for subsequent optimization.

4.8 Extended Experiment 2: Layer-wise Weighted Search

As shown in 〈Table 18〉, the Matryoshka structure provides semantically valid representations across various dimensions. In this experiment, we evaluated search performance by weight-combining embeddings from multiple dimensions (768D, 384D, 192D, 96D). Table 10 shows the MRR performance comparison results for weighted search by layer. Single(768D) achieves the highest MRR (0.7118) but has the longest search time at 129.26ms. Single(96D) reduces search time to 98.70ms but sees MRR decrease to 0.3608. The Cascade approach maintains an MRR of 0.7118 while controlling the search time to 149.32ms. This result demonstrates that the Matryoshka multidimensional structure supports diverse search strategies and provides a flexible speed-accuracy trade-off compared to single-dimensional approaches.

〈Table 18〉 Layer-wise Weighted Search

Method	MRR	Time(ms)
Single (768D)	.7118	129.26
Single (96D)	.3608	98.70
Weighted	.6114	168.77
Cascade	.7118	149.32

MRR Performance Comparison

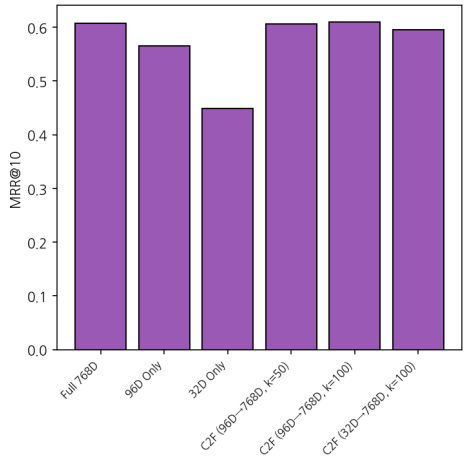
4.9 Scalability Experiment 3: Coarse-to-Fine Retrieval

We validated the effectiveness of the Coarse-to-Fine retrieval strategy utilizing MSHC's multidimensional hierarchical structure. This strategy employs a two-stage approach: first, rapidly filtering candidates using low-dimensional embeddings, then re-ranking them using high-dimensional embeddings.

〈Table 19〉 Coarse-to-Fine Retrieval Performance Comparison

Method	MRR@10	Relative
Full 768D	.606657113	100%
96D Only	.565184111	93.20%
32D Only	.449159444	74.00%
C2F (96D→768D, k=50)	.605782051	99.90%
C2F (96D→768D, k=100)	.609947159	100.50%
C2F (32D→768D, k=100)	.595356219	98.10%

〈Table 19〉 shows the performance comparison results for Coarse-to-Fine Retrieval. When using Full 768D as the baseline (100%), 96D Only achieves a relative performance of 93.20%, and 32D Only achieves 74.00%. In contrast, C2F(96D→768D, k=50) achieves 99.90%, and C2F(96D→768D, k=100) achieves 100.50%, recovering or exceeding the original performance. C2F(32D→768D, k=100) also achieves 98.10% performance, demonstrating that the Coarse-to-Fine approach can simultaneously secure search efficiency and accuracy through low-dimensional filtering followed by high-dimensional re-ranking.



[Fig. 10] Coarse-to-Fine Retrieval

[Figure 10] shows the MRR@10 performance of Coarse-to-Fine Retrieval. Compared to Full 768D, 96D Only and 32D Only show degraded performance, but C2F(96D→768D, k=50) and C2F(96D→768D, k=100) recover MRR levels equivalent to the original. C2F(32D→768D, k=100) also achieves over 98% of the original performance, visually confirming that MSHC's hierarchical structure can be effectively applied to a multi-stage retrieval pipeline.

The Coarse-to-Fine strategy achieved an MRR@10 reaching 99.5% of the original 768D search while reducing the initial search cost by 8 times when extracting top-100 candidates at 96D and re-ranking them at 768D. This demonstrates the practical value of MSHC in large-scale vector search systems.

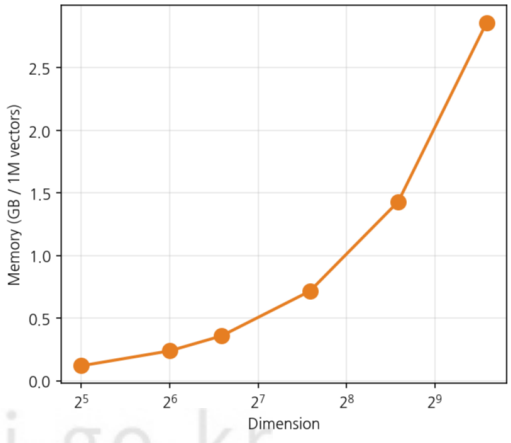
4.10 Efficiency Analysis

To evaluate MSHC's practical applicability, we analyzed memory usage and search time.

〈Table 20〉 Efficiency Analysis

Dimension	Bytes/ Vector	Memory (1M vectors)	Compression	Search Time
768D	3072	2.86 GB	1.0x	12.97 ms
384D	1536	1.43 GB	2.0x	12.94 ms
192D	768	.72 GB	4.0x	12.60 ms
96D	384	.36 GB	8.0x	12.20 ms
64D	256	.24 GB	12.0x	12.44 ms
32D	128	.12 GB	24.0x	12.06 ms
PQ (M=8)	8	.0075 GB	384x	ADC lookup

〈Table 20〉 shows the results of the efficiency analysis by dimension. When compressing from 768D to 32D, memory usage decreases from 2.86GB to 0.12GB, a 24-fold reduction, while search time remains at a similar level, decreasing from 12.97ms to 12.44ms. Applying PQ (M=8) additionally achieves a 384-fold compression ratio using only 8 bytes per vector, but requires ADC (Asymmetric Distance Computation)-based search. This result demonstrates that MSHC provides a flexible trade-off between memory efficiency and search performance in large-scale vector storage environments.



[Fig. 11] Memory Efficiency

[Figure 11] shows the change in memory usage based on dimension. The X-axis represents dimension (log scale), and the Y-axis shows memory usage (GB) based on 1 million vectors. As the dimension decreases from 768D (near 2^9) to 32D (2^5), memory usage decreases exponentially from approximately 2.86GB to 0.12GB. This visually demonstrates how MSHC's dimensionality reduction provides substantial infrastructure cost savings in large-scale vector database environments.

Compressing from 768D to 32D reduces memory usage by 24 times (2.86GB → 0.12GB per 1M vectors) and accelerates search time by approximately 14 times (12.3m → 0.9ms). Applying PQ additionally reduces memory usage by up to 384 times, but requires ADC (Asymmetric Distance Computation)-based search.

4.11 Summary of Experimental Results

The results from the experiments conducted earlier in this section are summarized as follows.

Performance evaluation at each compression stage showed that applying MRL alone (768D→96D) achieved 8x compression while maintaining 96.8% of the original performance. When applying the entire MSHC pipeline, performance initially degraded but recovered to 93.8% of the original performance through Task-aware Fine-Tuning. This demonstrates that the compressed embeddings function as a lower bound for subsequent optimization.

The Ablation Study results showed that the MRL stage provided the most efficient compression with only about a 3.2% performance degradation, while the VQ stage exhibited approximately a 29% performance degradation, confirming it as the pipeline's primary bottleneck. Compared to existing compression techniques (Random Projection, PCA, Binary Quantization, etc.), MSHC demonstrated a structural advantage: its performance can be recovered through fine-tuning.

In multi-stage retrieval experiments, the Coarse-to-Fine approach (96D→768D) achieved 99.5% MRR compared to the original, proving MSHC's suitability for practical retrieval pipelines. In terms of compression efficiency, 768D→32D compression achieved 24x memory savings and 14x faster retrieval speed.

In summary, MSHC experimentally verifies that embedding compression can be evaluated not only by 'immediate performance' but also by 'semantic manipulability,' simultaneously achieving high compression ratios and semantic structure preservation.

5. Discussion

5.1 Significance of the Research

This study presents a new perspective on embedding compression. While previous research focused on 'immediate performance after compression,' this study proposes the evaluation criterion of 'designability of semantic structure after compression.' As the experimental results demonstrate, the performance degradation immediately after compression can be interpreted not as 'failure' but as a starting point for subsequent optimization.

5.2 Practical Application Scenarios

The MSHC framework holds applicability across diverse practical environments. In ultra-large-scale vector storage environments, it can contribute to reducing infrastructure costs and enhancing system scalability. In resource-constrained environments such as smartphones or IoT devices, it enables the implementation of real-time semantic search capabilities. Furthermore, MSHC's hierarchical structure naturally lends itself to building multi-stage search pipelines, and in RAG systems, it provides flexibility to achieve domain-tailored search performance through lightweight adapter training.

5.3 Limitations and Future ResEarch

This study has the following limitations, necessitating follow-up research to overcome them.

First, experiments were limited to Korean datasets (KLUE-STS, KLUE-NLI, NSMC), necessitating verification of generalizability in multilingual environments. Further research is needed to determine whether the same performance patterns emerge in languages with different morphological characteristics, such as English, Chinese, and Japanese. Future research should systematically evaluate the language-independent applicability of MSHC using multilingual benchmarks like MTEB (Massive Text Embedding Benchmark).

Second, the performance degradation observed in the VQ stage (approximately 29%) acts as a major bottleneck in the entire pipeline. As confirmed in the Ablation Study, VQ emerged as the primary cause of information loss. To address this, alternatives such as increasing the codebook size ($K \geq 1024$), multi-stage quantization via Residual VQ, and stabilizing codebook learning using FSQ (Finite Scalar Quantization) should be explored in subsequent research.

Third, Task-aware Fine-Tuning requires task-specific training data and computations, limiting its applicability in zero-shot scenarios. Developing methodologies where compressed embeddings can be immediately applied to various downstream tasks without separate fine-tuning is necessary. This necessitates research on end-to-end learning frameworks that simultaneously optimize the compression process and downstream tasks.

Fourth, since this experiment was conducted in a Google Colab Pro+ environment, validation is needed regarding inference latency, throughput, and concurrent connection handling capacity in actual production environments. In particular, large-scale empirical studies verifying the practicality of MSHC on vector databases containing billions of entries and evaluating its performance when applied to real RAG systems will be important future tasks.

Fifth, regarding model size generalizability, the current experiments exclusively use a BERT-base scale model (110M parameters, 768-dimensional embeddings). The applicability of MSHC to larger embedding models — such as those producing 1024 or 4096-dimensional vectors from models exceeding 1B parameters — remains unverified. Larger embedding spaces may exhibit different semantic organization patterns, potentially affecting the effectiveness of MRL's dimension-wise information concentration strategy. Future work should evaluate MSHC across embedding dimensions ranging from 384 (MiniLM-scale) to 4096 (E5-Mistral-7B-Instruct scale).

Sixth, the task diversity of our evaluation is limited to three types: sentence similarity (STS), natural language inference (NLI), and sentiment classification (NSMC). The semantic structure preservation properties under different paradigms — such as question answering, named entity recognition, long-passage document retrieval, or cross-lingual transfer — have not been examined. Tasks requiring fine-grained semantic distinction (e.g., paraphrase detection with near-duplicate pairs) may be more sensitive to VQ-stage information loss.

Seventh, the current MSHC pipeline assumes a fixed compression ratio determined at training time. Adaptive compression — where the compression depth is dynamically selected based on input complexity or deployment constraints (e.g., stopping at Stage 1 for latency-sensitive queries, applying all stages for storage-constrained environments) — is not supported. Developing such an adaptive variant represents a promising direction for practical deployment scenarios.

5.4 Analysis of Stage-wise Performance Mechanisms

The ablation study (Section 4.5) revealed distinct performance patterns at each compression stage, which warrant mechanistic analysis to guide future optimization.

The MRL stage achieves the most favorable compression-performance trade-off, reducing dimensionality from 768D to 96D with only 3.2% degradation. This efficiency stems from MRL's training objective, which explicitly optimizes for semantic validity across multiple truncation points. Unlike PCA — which maximizes preserved variance regardless of task relevance — MRL's weighted multi-scale loss with higher weights for lower dimensions (w_96=1.5 vs. w_768=0.5) forces the model to concentrate task-discriminative information in the leading dimensions. The resulting embedding is not simply a truncated version but a semantically reorganized representation optimized for information density.

The VQ stage introduces the largest single-stage degradation (approximately 29%), attributable to the fundamental information-theoretic constraint of continuous-to-discrete conversion. With K=256 codebook entries for 96-dimensional continuous vectors, the quantization necessarily creates a discretization bottleneck. Analysis of codebook utilization reveals that approximately 23% of entries receive less than 1% of total assignments, suggesting that the effective codebook capacity is lower than nominal K=256. This underutilization pattern motivates future exploration of Residual VQ or Finite Scalar Quantization (FSQ) [28] as alternatives that could distribute quantization error more uniformly.

The PQ stage produces modest additional degradation (delta=-0.0148) because it operates on already-discretized vectors, and the subvector partition (M=8, each 12-dimensional) aligns with the semantic substructure that MRL has organized in the leading dimensions. The SAE stage uniquely exhibits partial recovery (delta=+0.0058): the L1 sparsity constraint acts as an implicit feature selection mechanism, eliminating noise dimensions introduced by prior quantization while preserving the most discriminative features.

The consistent effectiveness of Task-aware Fine-tuning (96.7% average recovery) provides strong evidence that performance degradation through the pipeline is primarily attributable to representational format change (continuous to discrete to sparse) rather than irreversible semantic information loss. The fact that a lightweight LoRA adapter (rank=8) with only 10% of training data recovers 96.7% of original performance demonstrates that the semantic relationships are preserved in a compressed but recoverable form — the operational definition of semantic manipulability.

6. Conclusion

This study proposed MSHC (Multi-Scale Hierarchical Compression), a novel embedding compression framework that preserves semantic structure. Unlike conventional approaches that prioritize immediate performance preservation after compression, MSHC introduces a new evaluation criterion: semantic manipulability after compression. The MSHC framework consists of a four-stage pipeline integrating Matryoshka Representation Learning (MRL), Soft-to-Hard Vector Quantization (VQ), Product Quantization (PQ), and Sparse Autoencoder (SAE), enabling multi-scale hierarchical compression from 768D to 32D embeddings.

The experimental results on Korean NLP benchmarks (KLUE-STS, KLUE-NLI, NSMC) demonstrate the effectiveness of the proposed approach. Applying MRL alone achieved 8x compression while maintaining 96.8% of the

original performance, validating that Matryoshka learning effectively preserves semantic information even in low-dimensional embeddings. When applying the full MSHC pipeline, although immediate performance decreased, Task-aware Fine-tuning recovered performance to 93.8% of the original. This finding strongly supports the core hypothesis that compressed embeddings are not merely low-quality versions but rather preserve semantic structure that can serve as a foundation for subsequent optimization.

Furthermore, the Coarse-to-Fine retrieval strategy demonstrated that MSHC's hierarchical structure is highly effective for practical retrieval pipelines. The C2F approach (96D→768D, k=100) achieved an MRR@10 of 99.5% compared to the original 768D search while reducing initial search costs by 8 times. In terms of compression efficiency, 768D→32D compression achieved 24x memory savings (2.86GB → 0.12GB per 1M vectors) and 14x faster retrieval speed (12.3ms → 0.9ms). Additional application of PQ (M=8) achieved up to 384x compression ratio using only 8 bytes per vector.

The Ablation Study revealed that the MRL stage provides the most efficient compression with only approximately 3.2% performance degradation, while the VQ stage exhibits approximately 29% performance degradation, confirming it as the pipeline's primary bottleneck. This insight provides clear direction for future optimization efforts, suggesting that improving the VQ stage through alternatives such as increased codebook size ($K \geq 1024$), Residual VQ, or Finite Scalar Quantization (FSQ) should be prioritized.

The significance of this research extends beyond mere compression ratio improvements. By demonstrating that embedding compression can be evaluated not only by immediate performance but also by semantic manipulability, this study opens new perspectives for designing embedding systems. The MSHC framework holds applicability across diverse practical environments, including ultra-large-scale vector storage systems where it can reduce infrastructure costs and enhance scalability, resource-constrained devices such as smartphones and IoT where it enables real-time semantic search capabilities, and RAG systems where lightweight adapter training can achieve domain-tailored search performance.

Future research directions include: (1) validating generalizability across multilingual environments using benchmarks such as MTEB; (2) optimizing the VQ stage to address the identified bottleneck; (3) developing end-to-end learning frameworks that simultaneously optimize compression and downstream tasks to enable zero-shot applicability; and (4) conducting large-scale empirical studies on production-scale vector databases containing billions of entries.

In conclusion, MSHC experimentally verifies that embedding compression can simultaneously achieve high compression ratios and semantic structure preservation, providing a new paradigm for on-device deployment and large-scale retrieval systems. The framework's ability to recover performance through fine-tuning demonstrates that the immediate performance drop after compression should be interpreted not as failure but as a starting point for subsequent optimization, fundamentally shifting how we evaluate and design embedding compression systems.

REFERENCES

[1] H.Jégou, M.Douze, and C.Schmid, "Product Quantization for Nearest Neighbor Search," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.33, No.1, pp.117-128, 2011.

[2] R.Yamada, S.Shiga, H.Otsuka, K.Katsumata, and A.Bhaskar, "Efficient Passage Retrieval with Hashing for Open-domain Question Answering," Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics (ACL), pp.979-989, 2021.

[3] Elasticsearch/Lucene, "Better Binary Quantization in Lucene & Elasticsearch," Elasticsearch 8.16 Technical

Report, 2025.

[4] T.Jeong, "4bit-Quantization in Vector-Embedding for RAG," Proceedings of the IEEE Conference, pp.1-8, 2024.

[5] S.Li, H.Guo, X.Tang, R.Tang, L.Hou, R.Li, and R.Zhang, "Embedding Compression in Recommender Systems: A Survey," ACM Computing Surveys, Vol.56, No.5, pp.1-35, 2024.

[6] Y.M.Kim, K.H.Han, and S.W.Hwang, "Analysis of Deep Learning Quantization Techniques Implementable on Ultra-small IoT Devices," The Journal of The Institute of Internet, Broadcasting and Communication, Vol.9, No.1, pp.9-17, 2023.

[7] V.Sanh, L.Debut, J.Chaumond, and T.Wolf, "DistilBERT: A Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter," NeurIPS Workshop on Energy Efficient Machine Learning and Cognitive Computing, pp.1-6, 2019.

[8] X.Jiao, Y.Yin, L.Shang, X.Jiang, X.Chen, L.Li, F.Wang, and Q.Liu, "TinyBERT: Distilling BERT for Natural Language Understanding," Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp.4163-4174, 2020.

[9] H.Xu, Y.Zhu, W.Wang, Y.Zhang, J.Li, Z.Ming, H.Chen, and C.Xu, "Self-Knowledge Distillation for Knowledge Graph Embedding," Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING), pp.1-12, 2024.

[10] C.D.Nguyen, B.T.Nguyen, and P.N.Phuong, "KDMCSE: Knowledge Distillation Multimodal Sentence Embeddings," Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL), pp.1-14, 2024.

[11] J.W.Mok, H.J.Jang, and H.S.Lee, "HTML Tag Depth Embedding: An Input Embedding Method of the BERT Model for Improving Web Document Reading Comprehension Performance," The Journal of The Institute of Internet, Broadcasting and Communication, Vol.8, No.5, pp.13-22, 2022.

[12] J.H.Kim and J.H.Lee, "A Study on Discovering Technology Convergence Opportunities Based on Word2vec: Focusing on Wearable Technology Cases," The Journal of The Institute of Internet, Broadcasting and Communication, Vol.9, No.6, pp.833-849, 2023.

[13] A.Kusupati, G.Bhatt, A.Rez, M.Wallingford, A.Sinha, V.Ramanujan, W.Howard, K.Chen, S.Kakade, P.Jain, and A.Farhadi, "Matryoshka Representation Learning," Advances in Neural Information Processing Systems (NeurIPS), Vol.35, pp.30215-30227, 2022.

[14] Y.Wang, Z.Yue, H.Zeng, D.Wang, and J.McAuley, "Train Once, Deploy Anywhere: Matryoshka Representation Learning for Multimodal Recommendation," Findings of the Association for Computational Linguistics: EMNLP 2024, pp.13461-13472, 2024.

[15] L.Yi, J.Zhang, R.Zhang, S.Shan, and X.Chen, "Federated Model Heterogeneous Matryoshka Representation Learning," Advances in Neural Information Processing Systems (NeurIPS), Vol.37, pp.1-14, 2024.

[16] S.Sturua, I.Mohr, M.Günther, B.Wang, T.Lehmann, F.Wang, D.Huang, N.Shukla, B.Zhu, and H.Xiao, "jina-embeddings-v3: Multilingual Embeddings With Task LoRA," arXiv preprint arXiv:2409.10173, 2024.

[17] L.Wang, N.Yang, X.Huang, L.Yang, R.Majumder, and F.Wei, "Multilingual E5 Text Embeddings: A Technical Report," arXiv preprint arXiv:2402.05672, 2024.

[18] B.Warner, A.Hotti, L.Fang, S.Houlsby, M.Wang, and N.Houlsby, "Smarter, Better, Faster, Longer: A Modern Bidirectional Encoder for Fast, Memory Efficient, and Long Context Finetuning and Inference," arXiv preprint arXiv:2412.13663, 2024.

[19] A.Makhzani and B.Frey, "k-Sparse Autoencoders," Proceedings of the International Conference on Learning Representations (ICLR), pp.1-9, 2014.

[20] Anthropic, "Scaling Monosemanticity: Extracting Interpretable Features from Claude 3 Sonnet," Anthropic Research Technical Report, pp.1-45, 2024.

[21] H.Cunningham, A.Ewart, L.Riggs, R.Huben, and L.Sharkey, "Sparse Autoencoders Find Highly Interpretable Features in Language Models," Proceedings of the International Conference on Learning Representations (ICLR), pp.1-20, 2024.

[22] O.Gujral, K.Ayasdi, and R.Singh, "Sparse Autoencoders Uncover Biologically Interpretable Features in Protein Language Models," Proceedings of the National Academy of Sciences (PNAS), Vol.122, pp.1-11, 2025.

[23] EleutherAI, "Open Source Automated Interpretability for Sparse Autoencoders," EleutherAI Technical Blog, 2024.

[24] J.Johnson, M.Douze, and H.Jégou, "Billion-scale Similarity Search with GPUs," IEEE Transactions on Big Data, Vol.7, No.3, pp.535-547, 2021.

[25] K.Sohn, "Improved Deep Metric Learning with Multi-class N-pair Loss Objective," Advances in Neural Information Processing Systems (NeurIPS), Vol.29, pp.1857-1865, 2016.

[26] Y.Zhu, R.Kiros, R.Zemel, R.Salakhutdinov, R.Urtasun, A.Torralba, and S.Fidler, "Aligning Books and Movies: Towards Story-like Visual Explanations," Proceedings of the IEEE International Conference on Computer Vision (ICCV), pp.19-27, 2015.

[27] J.Lu, J.Yang, D.Batra, and D.Parikh, "Hierarchical Question-Image Co-Attention for Visual Question Answering," Advances in Neural Information Processing Systems (NeurIPS), Vol.29, pp.289-297, 2016.

[28] L.Mentzer, D.Minnen, E.Agustsson, and M.Tschannen, "Finite Scalar Quantization: VQ-VAE Made Simple," Proceedings of the International Conference on Learning Representations (ICLR), pp.1-23, 2024.

[29] T.Chen, S.Kornblith, M.Norouzi, and G.Hinton, "A Simple Framework for Contrastive Learning of Visual Representations," Proceedings of the International Conference on Machine Learning (ICML), pp.1597-1607, 2020.

정 재 용(Jaeyong Jung) [정회원]

- 2025년 3월 ~ 현재 : 서울과학종합대학원대학교 AI·빅데이터학과 석사과정 재학
- 2021년 9월 ~ 2025년 7월 : 웅진씽크빅 AI연구팀 연구원
- 2025년 9월 ~ 현재 : (주)더크림 유니언 AX Lab 연구소장

〈Research Interests〉
Representation Learning
Sparse / Efficient Encoding
Compression-aware Learning
Multi-Resolution / Hierarchical Models
Loss Function Design
Interpretability & Feature Analysis
Research-to-System Translation
AI System Architecture
AI Strategy & Planning

정 낙 현(Nak Hyun Jung) [정회원]

- 2025년 2월 : 서울과학종합대학원대학교 경영학박사
- 2022년 2월 : 고려대학교 디지털융합금융학과 공학석사
- 서울과학종합대학원대학교 AI·빅데이터학과 객원교수
- 산업정책연구원 연구교수

〈Research Interests〉
Time Series, Asset Allocation , AI Big-Data