APPENDIX

A. Proof of Lemma 2

Proof. We consider the scenario where there exist two vertices u and v in the set of final neighbors $N(o)$ for a given vertex o, such that the angle between them, denoted as $\theta(u, o, v)$, is less than 60°. In the triangle $\triangle uov$, the sum of the angles $\theta(o, v, u)$ and $\theta(o, u, v)$ exceeds 120°. Here, the inner product (IP) of two vertices is used to measure the side length between them, and smaller IP values imply longer sides in the triangle. Therefore, we can conclude that either $\theta(o, v, u) > \theta(o, u, v)$ (i.e., $\theta(o, v, u) > 60^\circ$) or $\theta(o, u, v) > \theta(o, v, u)$ (i.e., $\theta(o, u, v) > 60^\circ$).

Case 1: If $\theta(o, v, u) > \theta(o, u, v)$, it follows that $IP(\hat{\boldsymbol{o}}, \hat{\boldsymbol{v}}) > IP(\hat{\boldsymbol{o}}, \hat{\boldsymbol{u}})$, indicating that the vertex v will be added to $N(o)$ before u (Line 14 in Algorithm 1). Since $\theta(u, o, v) < 60^\circ < \theta(o, v, u)$, we have $IP(\hat{\boldsymbol{u}}, \hat{\boldsymbol{v}}) > IP(\hat{\boldsymbol{o}}, \hat{\boldsymbol{u}})$. As a result, according to Line 16 in Algorithm 1, vertex u cannot be added to $N(o)$, which contradicts the initial assumption that u is in $N(o)$.

Case 2: If $\theta(o, u, v) > \theta(o, v, u)$, we can swap the positions of u and v in the triangle $\triangle uov$ and arrive at the same conclusion as in Case 1.

These cases demonstrate that the assumption of having two neighbors with an angle less than 60° in $N(o)$ is not feasible, and thus, it is ensured that the selected neighbors in $N(o)$ maintain an angle of at least 60° between each other, as described in Algorithm 1. This property ensures the effectiveness and correctness of the pipeline in constructing the final neighbor sets. □

B. Encoders Used in Our Experiments

ResNet. ResNet is a type of deep neural network that utilizes residual blocks and skip connections to facilitate the training of deep networks and mitigate the issue of vanishing or exploding gradients. It was proposed by researchers at Microsoft Research in 2015 [12] and achieved success in the ImageNet classification task with a 152-layer network. ResNet can also be applied to other visual recognition tasks, including object detection and segmentation. In our experiments, we employed ResNet17 and ResNet50 as encoders for the image modality. These are variations of ResNet with different numbers of layers. ResNet17 consists of 17 layers, while ResNet50 consists of 50 layers. Additionally, ResNet50 adopts a bottleneck design for its residual blocks, which reduces the parameter count and accelerates the training process. Both ResNet17 and ResNet50 can serve as feature extractors for tasks such as object detection or segmentation.

LSTM. LSTM stands for Long Short-Term Memory, which is a type of recurrent neural network (RNN) designed for processing sequential data, including speech and video [36]. LSTM incorporates feedback connections and a specialized structure called a cell, enabling it to store and update information over long time intervals. It also employs three gates (input, output, and forget) to regulate the flow of information into and out of the cell. LSTM finds applications in various tasks such as speech recognition, machine translation, and handwriting recognition. Furthermore, LSTM can be combined with convolutional neural networks (CNNs) to form a convolutional LSTM network, which proves useful for tasks like video prediction or object tracking. In our experiments, we utilized LSTM as the encoder for the text modality.

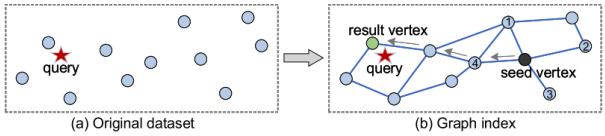
Transformer. The Transformer is a deep learning model introduced in 2017 for natural language processing tasks, such as machine translation and text summarization [11]. Unlike recurrent neural networks, the Transformer does not process sequential data in a sequential manner. Instead, it employs attention mechanisms to capture dependencies between words or tokens. The Transformer comprises two main components: an encoder and a decoder. The encoder takes an input sentence and converts it into a sequence of vectors known as encodings. The decoder takes these encodings and generates an output sentence. Both the encoder and decoder consist of multiple layers, each containing a multi-head self-attention module and a feed-forward neural network module. Additionally, the Transformer utilizes positional encodings to incorporate positional information for each word in the sentence. In our experiments, we also employed the Transformer to encode the text modality.

GRU. A Gated Recurrent Unit (GRU) encoder is a type of recurrent neural network (RNN) that can encode input sequences of varying lengths into fixed-length feature vectors [55]. It selectively updates and resets its hidden state based on the input and previous state, allowing it to capture both short-term and long-term dependencies within the sequence. This concise representation generated by the GRU encoder can be utilized for various tasks, including machine translation, speech recognition, and natural language understanding. In our experiments, we employed the GRU encoder to encode the text modality.

Encoding. In our experiments, we utilized ordinal encoding [35] to encode the structured text description. This technique transforms categorical data into numerical data by assigning integer values to categories based on their rank or order. For instance, if a feature has three categories: "low", "medium", and "high", they can be encoded as 1, 2, and 3 respectively. Ordinal encoding is appropriate for categorical features that possess a natural ordering, such as grades, sizes, ratings, and so on. The original categories can be restored by reversing the ordinal encoding process, which involves mapping the integer values back to their respective categories.

TIRG. TIRG, which stands for Text-Image Residual Gating, is a method used to merge image and text features for image retrieval tasks [7]. It involves modifying the features of the query image using text, while maintaining the resulting feature vector within the same space as the target image. This is accomplished through a gated residual connection, which enhances the encoding and learning of representations. In our experiments, we utilized TIRG to encode image-text pairs as composition vectors.

CLIP. CLIP, which stands for Contrastive Language-Image



(a) Original dataset (b) Graph index

Fig. 12: An example of indexing and search based on proximity graph [23].

Pre-Training, is a neural network model developed by OpenAI that has demonstrated remarkable achievements in multimodal zero-shot learning [13]. It is trained on a large dataset of image-text pairs collected from the web and learns to associate images with their corresponding textual descriptions. CLIP exhibits impressive generalization capabilities and has been successfully applied to various tasks, including fine-grained art classification, image generation, zero-shot video retrieval, event classification, and visual commonsense reasoning. In our experiments, we also employed CLIP to encode image-text pairs as composition vectors.

MPC. MPC, which stands for Multimodal Probabilistic Composer, is a model designed to encode multiple modalities from diverse visual and textual sources [46]. It utilizes a probabilistic rule to combine probabilistic embeddings and employs a probabilistic similarity metric to measure the distance between them. The functioning of MPC is as follows: given information from various visual or textual modalities, it first learns probabilistic embeddings for each modality. These embeddings are then merged using a probabilistic composer, resulting in a probabilistic compositional embedding. This embedding is subsequently matched with the probabilistic embedding of the desired image by minimizing a probabilistic distance metric. MPC is capable of processing more than two queries by applying the probabilistic composer to a set of probabilistic embeddings. In our experiments, we employed MPC to encode image-text-image triples as composition vectors.

C. High-dimensional Vector Search

Vector search is a fundamental task with applications across various domains [41], [56], and it has received significant attention in recent years due to advancements in representation learning methods [24], [57]. However, exact vector search can be computationally expensive, prompting researchers to focus on developing approximate techniques that strike a balance between accuracy and efficiency using vector indexes [58], [59], [60], [61]. Current vector search methods can be categorized into four types based on how the index is constructed: tree-based methods [62], [63], [64], quantization-based methods [59], [65], [66], hashing-based methods [67], [68], [69], and proximity graph-based methods [24], [25], [39]. Recent works [41], [39] have demonstrated that proximity graph-based methods achieve a favorable trade-off between accuracy and efficiency, making them well-suited for handling large-scale vector search tasks.

D. Proximity Graph-Based Index Algorithm

Proximity graph-based algorithms have gained popularity for vector similarity search, particularly in high-dimensional spaces, due to their ability to strike a balance between efficiency and accuracy by capturing neighbor relationships between vectors [25]. Major high-tech companies like Microsoft [51] and Alibaba [25] utilize these algorithms. To enable online query serving, an offline proximity graph index needs to be built on the dataset of feature vectors. This graph consists of vertices representing the vector data points and edges representing pairwise similarities or distances between vectors. Different algorithms, such as NSG [25] or KGraph [40], employ various graph construction methods.

In a recent survey [23], a comprehensive analysis of proximity graph-based index algorithms is provided, including their performance, strengths, and potential pitfalls. Fig. 12 illustrates an example of finding the nearest vertex to a query vector q using a proximity graph index. The process begins with a seed vertex (the black vertex), which can be randomly selected or fixed [23]. It then visits its neighbors and computes their distances to q. Vertex 4 is chosen as the next visiting vertex because it is the closest among the seed's neighbors. This process continues until it reaches the green vertex, which has no neighbors closer to q than itself. The search process relies on various factors, such as the seed acquisition strategy [60] and the routing technique [56].

It is worth noting the following remarks:
(1) Lack of Theoretical Guarantee. Although state-of-the-art proximity graph index algorithms lack theoretical guarantees [23], [24], their superiority in real-world scenarios has been validated by numerous research works [23], [24], [25], [41], [60], [39], [51] and industrial applications [70], [71].
(2) Flexibility and Customization in MUST. In MUST, we have designed a general pipeline that allows components from existing proximity graph algorithms to be easily integrated. Moreover, our pipeline supports custom-optimized components, which can inspire further research and experimentation.

Overall, proximity graph-based indexes provide an effective solution for vector similarity search, and their practical performance has been well-established in real-world scenarios.

E. Motivation for Vector Weight Learning

In our framework, we aim to combine m vectors of an object with m modalities by assigning weights to each vector, resulting in a concatenated vector. This concept is inspired by similar cases, such as calculating multi-metric distance in multi-metric spaces [72]. However, determining the importance or relevance of different vectors is a challenging task. Current methods often rely on user-defined weights [72], [73], which has two limitations.
(1) Lack of User-friendliness. Assigning proper weights to different vectors is not user-friendly since users may not have the necessary knowledge or understanding to determine appropriate weights. This manual weight assignment process can be subjective and may not reflect the true importance of each vector. Based on our experiments, we have observed that different weights significantly affect the recall rate of MSTM, as shown in Fig. 9 of the main text.
(2) Inapplicability for Offline Index Construction. To build a fused index, as proposed in our paper, the weights for

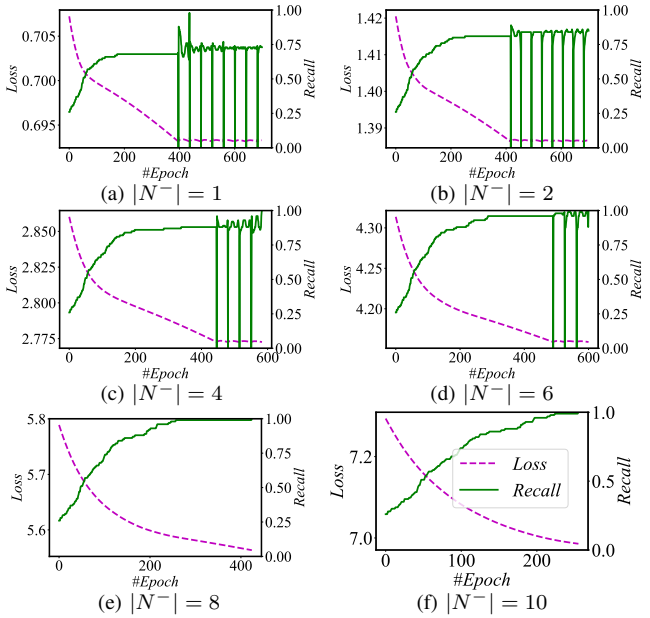


Fig. 13: Effect of different number of negatives in vector weight learning.

measuring the similarity between objects need to be known in advance during the offline index construction phase. However, relying on user-defined weights obtained online is not suitable for this purpose. Online weight assignment may introduce inconsistency and hinder the offline index construction process, which requires a consistent set of weights.

To address these limitations, there is a need for an automated approach to learn the vector weights that overcomes the user-defined weight assignment challenge and enables efficient offline index construction. By automatically learning the weights, we can ensure that the similarity computation process captures the true relevance and contribution of each vector, making it more objective and reliable.

F. More Details of Setup and Parameters

We adopt the same training hyperparameters as the original papers of the encoders to obtain the embedding vectors. The encoder configuration remains consistent across all three frameworks. Our training pipeline is implemented in PyTorch for the vector weight learning model, and we utilize the Pybind library to invoke the vector similarity search kernel written in C++. The learning rate was set to 0.002, and the training was conducted for 700 iterations by default.

The indexing components and search codes were implemented in C++ using CGraph [38] and compiled with g++6.5. We built all indexes in parallel using 64 threads and executed the search procedure using a single thread, following the common setting in related work [23], [25].

All experiments were conducted on a Linux server equipped with an Intel(R) Xeon(R) Gold 6248R CPU running at 3.00GHz and 755G memory. We performed three repeated trials and reported the average results for all evaluation metrics. The learned weights can be found in Appendix K [30].

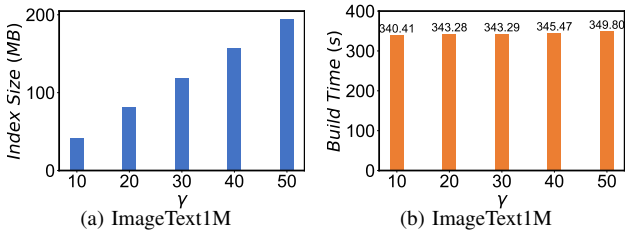


Fig. 14: Index size and index build time under different values of γ.

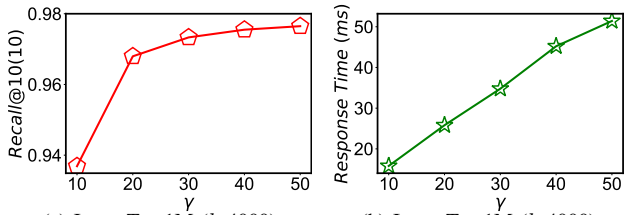


Fig. 15: Search performance under different values of γ.

G. Study of the Number of Negative Examples

In our study, we investigate the impact of the number of negative examples (e.g., $|N^-|$) on model training. We evaluate this effect by examining the loss and recall rate curves on the hard negatives using the ImageText1M dataset. As shown in Fig. 13, we observe that increasing the number of negative examples generally leads to better training results. This means that including more negative examples during training helps improve the model's performance in identifying positive examples and distinguishing them from hard negatives.

However, it is important to consider the trade-off between training quality and training efficiency. As the number of negative examples increases, the training time also tends to increase. Therefore, it is necessary to find a proper balance between the training quality and efficiency by selecting an appropriate value for $|N^-|$.

By analyzing the loss and recall rate curves for different values of $|N^-|$, we can determine the optimal number of negative examples that achieves a satisfactory training effect without excessively increasing the training time. This ensures that the model is trained effectively while taking into account practical considerations.

H. Parameters for Fused Index

The construction of the fused index relies on two important parameters: the maximum number of neighbors (γ) and the maximum iterations (ε). These parameters have an impact on the index size, index build time, and search performance. In this section, we evaluate these parameters in more detail.

Maximum Number of Neighbors (γ). Fig. 14 illustrates the relationship between γ and the index size as well as the index build time. As γ increases, both the index size and the index build time also increase. This is because a larger γ requires handling more vertices in the *Initialization*, *Candidate acquisition*, and *Neighbor selection* components, which increases the complexity of index construction.

In Fig. 15, we maintain the other parameters unchanged (including the search parameters k and l) and analyze the search performance for different values of γ. The result shows

TABLE XI: Graph quality under different number of iterations.

# Iterations (ε) $\downarrow$	ImageText1M	AudioText1M	VideoText1M
1	0.0094	0.0088	0.0096
2	0.7795	0.7945	0.7842
3	0.9900	0.9900	0.9900

TABLE XII: Search performance under different values of l ($\gamma = 30$).

$l \rightarrow$	700	1000	1500	2000	4000
$\boldsymbol{Recall@10(10)}$	0.506100	0.637260	0.766190	0.856250	0.973310
Response Time (ms)	5	7	11	15	35

that the recall rate improves as γ increases. This is because a larger γ allows for visiting more neighbors of a vertex during the search process, leading to improved search accuracy. However, when γ becomes very large, the computation of vector distances increases, resulting in lower efficiency. Therefore, it is crucial to strike a balance between accuracy and efficiency by adjusting γ in practical scenarios. In our experiments, we set γ to a default value of 30.

Maximum iterations (ε). The graph quality is defined as the mean ratio of γ neighbors of a vertex over the top-γ nearest neighbors based on joint similarity [23]. Tab. XI presents the changes in graph quality with different numbers of iterations (ε). The results demonstrate that graph quality increases with ε and reaches a value close to 1 when ε is set to 3. Therefore, we set ε to a default value of 3 in all our experiments.

By evaluating these parameters, we gain insights into their impact on the fused index construction. The findings suggest that selecting appropriate values for γ and ε can optimize index size, build time, and search performance, striking a balance between accuracy and efficiency in practice.

I. Parameter for Joint Search

The parameter that affects the accuracy and efficiency of the search process in the joint search algorithm (please refer to Algorithm 2 of the main text) is the result set size, denoted as l. In this section, we evaluate how the recall rate and response time change with different values of l.

Tab. XII presents the results of the evaluation, showing the recall rate and response time for different l values. It can be observed that both the recall rate and response time increase as l becomes larger. The increase in recall rate is expected because a larger l allows for visiting more vertices and considering more potential results during the search process. This leads to a higher likelihood of retrieving relevant objects, resulting in an improved recall rate. However, the response time also increases with larger l values. This is because a larger result set size requires visiting more vertices and performing more vector calculations, which adds computational overhead and increases the overall response time. Therefore, selecting an appropriate value for l involves a trade-off between recall rate and response time. A larger l can improve recall but at the cost of increased response time. Researchers and practitioners need to consider the specific requirements of their application and strike a balance between accuracy and efficiency when choosing the value of l.

J. Datasets

We utilize nine datasets with varying modalities and cardinalities obtained from public sources, as presented in Tab. II of the main text. CelebA [34], MIT-States [44], Shopping [45], and MS-COCO [46] are four real-world multimodal datasets [3], [7]. For instance, CelebA comprises two modalities for each object: a facial image and a corresponding text description. In these datasets, we employ the original query samples, and each query contains one or more ground-truth objects. Since there are no publicly available datasets with up to four modalities, we simulated two additional modalities for CelebA using different encoders. This led to the creation of the CelebA+ dataset, with four vectors for each object, simulating four modalities. To evaluate performance at a large scale, we added the text modality to four single-modal datasets (DEEP and SIFT for images [74], [47], MSONG for audio [48], and UQ-V for video [49]) using the same method described in [32]. As a result, we formed four large-scale multimodal datasets: ImageText16M, ImageText1M, AudioText1M, and VideoText1M.

CelebA. CelebFaces Attributes Dataset (CelebA) [34] is a comprehensive dataset of celebrity images, consisting of over 200,000 images. Each image is annotated with 40 attributes, 5 landmark locations, and a face identity. The dataset covers a wide range of poses and backgrounds, offering rich diversity, quantity, and annotations. CelebA can be utilized for various computer vision tasks, including face attribute recognition, face recognition, face detection, landmark localization, and face editing & synthesis.

MIT-States. MIT-States [44] is a dataset comprising approximately 60,000 images, each labeled with an object/noun and a state/adjective (e.g., "red tomato" or "new camera"). The dataset includes 245 nouns and 115 adjectives, with an average of around 9 adjectives per noun. MIT-States is commonly employed to evaluate image retrieval and image classification tasks in the field of computer vision.

Shopping. The Shopping100k dataset [45] consists of 101,021 images of clothing items extracted from various e-commerce providers for fashion studies. It was developed to address limitations in existing fashion-related datasets, which often feature posed images with occlusion issues. Each image in the dataset is represented with general and special attributes, with the special attributes being more suitable for attribute manipulation and fashion searches. Shopping encompasses various categories, such as T-shirts and bottoms.

MS-COCO. The MS-COCO dataset [46] is a widely used dataset in computer vision research, standing for Microsoft Common Objects in Context. It comprises over 330,000 images with more than 2.5 million labeled object instances, annotated with object bounding boxes and belonging to 80 object categories. The dataset is designed to facilitate research on object detection, segmentation, captioning, and other related tasks, serving as a challenging benchmark for evaluating computer vision models due to its scale, diversity, and complexity.

ImageText1M. ImageText1M is a semi-synthetic dataset that combines real-world images with text. It consists of 1 million SIFT vectors with a dimension of 128 [47]. Each vector

TABLE XIII: Output weights of module for MIT-States dataset.

Encoder ↓	ω_0^2 (modality 0)	ω_1^2 (modality 1)
ResNet17+LSTM	0.3000	0.7000
ResNet50+LSTM	0.0012	1.4291
ResNet17+Transformer	0.1172	0.2669
ResNet50+Transformer	0.5000	0.5000
TIRG+LSTM	0.5000	0.5000
TIRG+Transformer	0.0295	0.0224
CLIP+LSTM	0.5000	0.5000
CLIP+Transformer	0.0670	0.0432

TABLE XIV: Output weights of module for CelebA dataset.

Encoder ↓	ω_0^2 (modality 0)	ω_1^2 (modality 1)
ResNet17+Encoding	0.0007	0.9526
ResNet50+Encoding	0.0848	1.1855
TIRG+Encoding	0.1064	0.6414
CLIP+Encoding	0.1089	0.8551

TABLE XV: Output weights of module for Shopping dataset.

Encoder ↓	ω_0^2 (modality 0)	ω_1^2 (modality 1)
ResNet17+Encoding	0.0262	1.2124
TIRG+Encoding	0.0092	1.2042

TABLE XVI: Output weights of module for MS-COCO dataset.

Encoder ↓	ω_0^2	ω_1^2	ω_2^2
MPC+GRU+ResNet50	0.0083	0.0342	0.0123
ResNet50+GRU+ResNet50	0.0091	0.0233	0.0144

TABLE XVII: Output weights of module for CelebA+ dataset.

Encoder ↓	ω_0^2	ω_1^2	ω_2^2	ω_3^2
CLIP+Encoding+ResNet17+ResNet50	0.4092	3.1363	0.0721	0.0290

TABLE XVIII: Output weights of module for ImageText1M, AudioText1M, VideoText1M, and ImageText16M datasets.

Dataset ↓	ω_0^2 (modality 0)	ω_1^2 (modality 1)
ImageText1M	0.1199	0.5572
AudioText1M	0.0453	0.8589
VideoText1M	0.3106	0.4440
ImageText16M	0.1123	0.8742

represents an image and is augmented with a text modality to form a multimodal dataset.

AudioText1M. AudioText1M is a semi-synthetic dataset that combines real-world audio with text. It comprises 1 million contemporary popular music tracks, each represented by 420 dimensions of audio features and metadata [48]. Similar to ImageText1M, each audio vector is paired with a text modality to create a multimodal dataset.

VideoText1M. VideoText1M is a semi-synthetic dataset that combines real-world videos with text. It extracts 256 dimensions of local features from keyframes of each video [49]. These video vectors are then combined with a text modality, resulting in a multimodal dataset.

ImageText16M. ImageText16M is a semi-synthetic dataset that merges real-world images with text. It encompasses 16 million data points, with each point represented by 96 dimensions of deep neural codes derived from a convolutional neural network [74]. Similar to the other multimodal datasets, a text modality is added to each image vector.

K. Weights Setting

In MUST, we employ a vector weight learning module to capture the significance of various modalities. The specific weights utilized for constructing indexes and performing query processing on different datasets and encoders are presented in Tab. XIII to XVIII.

TABLE XIX: Search accuracy with target modality input only.

Dataset	**Encoder**	***Recall@1(1)***	***Recall@5(1)***	***Recall@10(1)***
MIT-States	ResNet17	0.0268	0.1103	0.1822
	ResNet50	0.0363	0.1393	0.2257
CelebA	ResNet17	0.1499	0.4055	0.4913
	ResNet50	0.1475	0.3785	0.4519
Shopping (T-shirt)	ResNet17	0	0.0192	0.0399

TABLE XX: Search accuracy with auxiliary modality only.

Dataset	**Encoder**	**Recall@1(1)**	**Recall@5(1)**	**Recall@10(1)**
MIT-States	LSTM	0.2747	0.4343	0.4844
	Transformer	0.2601	0.2641	0.2824
CelebA	Encoding	0.0377	0.0936	0.1291
Shopping (T-shirt)	Encoding	0.0964	0.4126	0.5362

TABLE XXI: Search accuracy on Shopping (Bottoms).

Framework	**Encoder**	**Recall@1(1)**	**Recall@5(1)**	**Recall@10(1)**
JE	TIRG	0.0905	0.2715	0.3924
MR	ResNet17+Encoding	0.0107	0.0551	0.0995
	TIRG+Encoding	0.0596	0.2552	0.3850
MUST (ours)	ResNet17+Encoding	**0.4840(↑434.8%)**	0.7960	0.8887
	TIRG+Encoding	0.4784	**0.8162(↑200.6%)**	**0.8999(↑129.3%)**

L. Search Accuracy Using a Single Modality

In this section, we analyze the search accuracy achieved when using only the target modality input or the auxiliary modality input. Tab. XIX presents the search accuracy using only the target modality input, while Tab. XX displays the search accuracy using only the auxiliary modality input, both evaluated on three real-world datasets. Generally, these unimodal approaches exhibit lower performance compared to methods that combine multiple modalities. However, in certain cases, they outperform the JE framework, which combines the features of all modality inputs into a joint embedding. One possible explanation for this observation is that the JE framework introduces a significant encoder error, while the auxiliary modality can accurately describe an object in some datasets. This finding further emphasizes the necessity of utilizing multiple vectors from different encoders to effectively represent an object.

M. Search Accuracy on Shopping (Bottoms)

In this section, we present the search accuracy specifically for the "bottoms" category of the Shopping dataset, as shown in Tab. XXI. It is worth noting that different categories within the Shopping dataset share the same output weights. This observation highlights the generalization capability of the vector weight learning module employed in our framework.

N. Discussion of Accuracy, Efficiency, and Scalability

Accuracy. In the context of MSTM, unimodal search results tend to be inaccurate. Two baseline methods, namely Joint Embedding (JE) and Multi-streamed Retrieval (MR), aim to improve accuracy by incorporating multimodal information. JE creates a composition vector by combining different modalities, while MR merges the results of multiple unimodal searches. One way to optimize these baselines is to use JE as one of the separate search methods within MR. However, this optimization is still limited by MR's inability to capture the importance of different modalities. MUST addresses this limitation by employing a hybrid fusion mechanism that combines modalities at multiple levels. By utilizing a weight-learning module, it effectively captures the importance of different modalities and achieves the highest accuracy.

Efficiency. Performing MSTM involves computationally intensive tasks such as building an index and performing approximate queries. MR builds an index for each modality and searches them separately, which can be slow in large-scale scenarios, especially when the number of query results increases. MUST achieves higher efficiency and recall rate by constructing a fused index for different modal vectors and performing joint searches. The main reason for the efficiency difference lies in the quality of the graph index. In the main text, Fig. 11 provides a visualization of three neighbors for an object in the CelebA dataset. In MUST's index, the vertex and its neighbors balance the importance of different modalities and exhibit better joint similarity. On the other hand, in MR's indexes, the vertex and its neighbors only consider similarity within a single modality.

Scalability. As the number of modalities and the data scale increase, MR requires more and larger indexes, resulting in high storage costs and low indexing efficiency. In contrast, the size of MUST's index and its construction time grow slightly (almost logarithmically) with the data size and remain the same regardless of the number of modalities. It is important to note that as more data is added, the merging operation in MR becomes more complex, limiting its search efficiency and accuracy. MUST proves to be more efficient and accurate in large-scale scenarios and does not require a merging operation. Additionally, MUST handles multiple modalities more effectively than MR.

Highlight. We introduce the concept of modality importance mining in the context of MSTM, thereby opening up new research avenues in multimodal search. We anticipate that this will drive advancements in representation learning, vector indexing, and search algorithms.

O. Result Examples on Real-world Datasets

Fig. 16 shows the top-10 search results obtained using different frameworks for a query input consisting of a clock image and the text description "change state to melted". The results demonstrate that MUST successfully retrieves all the ground-truth objects. However, MR and JE returns many unrelated objects. This discrepancy occurs because MR combines dissimilar objects with low ranks from each candidate set, and the high encoding error of JE leads to a larger inner product (IP) between the query input's composition vector and the vectors of dissimilar objects in the target modality.

Fig. 17 displays the top-10 search results obtained using different frameworks for a query input consisting of a fresh cheese image and the text description "change state to moldy". The results indicate that MUST successfully recalls all the ground-truth objects, whereas MR and JE only retrieve a few ground-truth objects, and many of the results only match the text description.

Fig. 18 showcases the top-10 search results obtained using different frameworks for a query input consisting of a male face image and the text description "change state to bags under eyes, high cheekbones, mouth slightly open, and smiling". It is important to note that in addition to matching the query input face and text description, we also require that the result face and the query input face belong to the same identity. The results demonstrate that MUST retrieves more ground-truth faces compared to other frameworks.

Fig. 18 presents the top-10 search results obtained using different frameworks for a query input consisting of a female face image and the text description "change state to arched eyebrows and pointy nose". Similar to the previous example, we require that the result face and the query input face belong to the same identity. This requirement makes it more challenging to obtain the target face. Nevertheless, the results demonstrate that MUST retrieves more ground-truth faces compared to other frameworks.

Fig. 20 illustrates the top-10 search results obtained using different frameworks for a query input consisting of a T-shirt image and the text description "replace gray color with white color and replace sweat fabric with jersey fabric". The results indicate that MUST retrieves more ground-truth T-shirts compared to other frameworks.

Fig. 21 displays the top-10 search results obtained using different frameworks for a query input consisting of a T-shirt image and the text description "replace sweat fabric with jersey fabric and replace striped pattern with print pattern". The results demonstrate that MUST retrieves more ground-truth T-shirts compared to other frameworks.

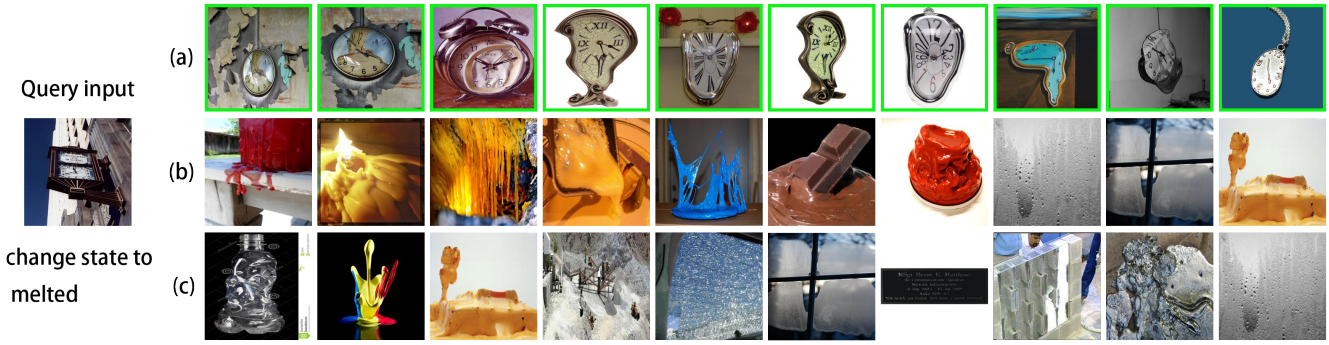


Fig. 16: Some result examples of different frameworks with optimal encoders on MIT-States dataset. (a), (b), and (c) are the top-10 results of MUST, MR, and JE, respectively. The green box marks the target objects.

Fig. 17: Some result examples of different frameworks with optimal encoders on MIT-States dataset. (a), (b), and (c) are the top-10 results of MUST, MR, and JE, respectively. The green box marks the target objects.

Fig. 18: Some result examples of different frameworks with optimal encoders on CelebA dataset. (a), (b), and (c) are the top-10 results of MUST, MR, and JE, respectively. The green box marks the target objects.

Fig. 19: Some result examples of different frameworks with optimal encoders on CelebA dataset. (a), (b), and (c) are the top-10 results of MUST, MR, and JE, respectively. The green box marks the target objects.

Fig. 20: Some result examples of different frameworks with optimal encoders on Shopping dataset. (a), (b), and (c) are the top-10 results of MUST, MR, and JE, respectively. The green box marks the target objects.

Fig. 21: Some result examples of different frameworks with optimal encoders on Shopping dataset. (a), (b), and (c) are the top-10 results of MUST, MR, and JE, respectively. The green box marks the target objects.

REFERENCES

[1] I. Tautkute, T. Trzciński, A. P. Skorupa, Ł. Brocki, and K. Marasek, "Deepstyle: Multimodal search engine for fashion and interior design," *IEEE Access*, vol. 7, pp. 84 613–84 628, 2019.

[2] J. Etzold, A. Brousseau, P. Grimm, and T. Steiner, "Context-aware querying for multimodal search engines," 2012.

[3] S. Jandial, P. Badjatiya, P. Chawla, A. Chopra, M. Sarkar, and B. Krishnamurthy, "Sac: Semantic attention composition for text-conditioned image retrieval," in *CVPR*, 2022, pp. 4021–4030.

[4] H. Wen, X. Song, X. Yang, Y. Zhan, and L. Nie, "Comprehensive linguistic-visual composition network for image retrieval," in *SIGIR*, 2021, pp. 1369–1378.

[5] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval: The Concepts and Technology behind Search*, 2nd ed. USA: Addison-Wesley Publishing Company, 2011.

[6] T. Baltrušaitis, C. Ahuja, and L.-P. Morency, "Multimodal machine learning: A survey and taxonomy," *TPAMI*, vol. 41, no. 2, pp. 423–443, 2018.

[7] N. Vo, L. Jiang, C. Sun, K. Murphy, L.-J. Li, L. Fei-Fei, and J. Hays, "Composing text and image for image retrieval-an empirical odyssey," in *CVPR*, 2019, pp. 6439–6448.

[8] M. Patel, T. Gokhale, C. Baral, and Y. Yang, "CRIPP-VQA: counterfactual reasoning about implicit physical properties via video question answering," in *EMNLP*, 2022, pp. 9856–9870.

[9] "Mum brings multimodal search to lens, deeper understanding of videos and new serp features," https://searchengineland.com/mum-brings-multimodal-search-to-lens-deeper-understanding-of-videos-and-new-serp-features-374798, 2021.

[10] T. Yu, Y. Yang, Y. Li, L. Liu, M. Sun, and P. Li, "Multi-modal dictionary bert for cross-modal video search in baidu advertising," in *CIKM*, 2021, pp. 4341–4351.

[11] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *NAACL*, 2019, pp. 4171–4186.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[13] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *ICML*, 2021, pp. 8748–8763.

[14] S. Liu, L. Li, J. Song, Y. Yang, and X. Zeng, "Multimodal pre-training with self-distillation for product understanding in e-commerce," in *WSDM*, 2023, p. 1039–1047.

[15] M. Levy, R. Ben-Ari, N. Darshan, and D. Lischinski, "Chatting makes perfect–chat-based image retrieval," *arXiv:2305.20062*, 2023.

[16] V. Ramanishka, "Describing and retrieving visual content using natural language," Ph.D. dissertation, Boston University, 2020.

[17] J. Ma, Y. Chen, F. Wu, X. Ji, and Y. Ding, "Multimodal reinforcement learning with effective state representation learning," in *AAMAS*, 2022, pp. 1684–1686.

[18] A. Majumdar, A. Shrivastava, S. Lee, P. Anderson, D. Parikh, and D. Batra, "Improving vision-and-language navigation with image-text pairs from the web," in *ECCV*, 2020, pp. 259–274.

[19] Y. Zhao, Y. Song, and Q. Jin, "Progressive learning for image retrieval with hybrid-modality queries," in *SIGIR*, 2022, pp. 1012–1021.

[20] K. Zagoris, A. Arampatzis, and S. A. Chatzichristofis, "www. mmretrieval. net: a multimodal search engine," in *SISAP*, 2010, pp. 117–118.

[21] C. Wei, B. Wu, S. Wang, R. Lou, C. Zhan, F. Li, and Y. Cai, "Analyticdb-v: A hybrid analytical engine towards query fusion for structured and unstructured data," *PVLDB*, vol. 13, no. 12, pp. 3152–3165, 2020.

[22] S. Zhang, M. Yang, T. Cour, K. Yu, and D. N. Metaxas, "Query specific rank fusion for image retrieval," *TPAMI*, vol. 37, no. 4, pp. 803–815, 2014.

[23] M. Wang, X. Xu, Q. Yue, and Y. Wang, "A comprehensive survey and experimental comparison of graph-based approximate nearest neighbor search," *PVLDB*, vol. 14, no. 11, pp. 1964–1978, 2021.

[24] Y. A. Malkov and D. A. Yashunin, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs," *TPAMI*, vol. 42, no. 4, pp. 824–836, 2020.

[25] C. Fu, C. Xiang, C. Wang, and D. Cai, "Fast approximate nearest neighbor search with the navigating spreading-out graph," *PVLDB*, vol. 12, no. 5, pp. 461–474, 2019.

[26] F. Zhang, M. Xu, Q. Mao, and C. Xu, "Joint attribute manipulation and modality alignment learning for composing text and image to image retrieval," in *ACM MM*, 2020, pp. 3367–3376.

[27] G. Delmas, R. S. de Rezende, G. Csurka, and D. Larlus, "ARTEMIS: attention-based retrieval with text-explicit matching and implicit similarity," in *ICLR*, 2022.

[28] A. Baldrati, M. Bertini, T. Uricchio, and A. Del Bimbo, "Conditioned and composed image retrieval combining and partially fine-tuning clip-based features," in *CVPR*, 2022, pp. 4959–4968.

[29] "Revolutionizing semantic search with multi-vector hnsw indexing in vespa," https://blog.vespa.ai/semantic-search-with-multi-vector-indexing/, 2023.

[30] M. Wang, X. Ke, X. Xu, L. Chen, Y. Gao, P. Huang, and R. Zhu, "Must: An effective and scalable framework for multimodal search of target modality," https://github.com/ZJU-DAILY/MUST/blob/main/appendix/MUST-appendix.pdf, 2023.

[31] L. Kennedy, S.-F. Chang, and A. Natsev, "Query-adaptive fusion for multimodal search," *Proceedings of the IEEE*, vol. 96, no. 4, pp. 567–588, 2008.

[32] J. Wang, X. Yi, R. Guo, H. Jin, P. Xu, S. Li, X. Wang, X. Guo, C. Li, X. Xu, K. Yu, Y. Yuan, Y. Zou, J. Long, Y. Cai, Z. Li, Z. Zhang, Y. Mo, J. Gu, R. Jiang, Y. Wei, and C. Xie, "Milvus: A purpose-built vector data management system," in *SIGMOD*, 2021, pp. 2614–2627.

[33] M. Wang, L. Lv, X. Xu, Y. Wang, Q. Yue, and J. Ni, "Navigable proximity graph-driven native hybrid queries with structured and unstructured constraints," *arXiv:2203.13601*, 2022.

[34] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *ICCV*, 2015, pp. 3730–3738.

[35] Z. Wang, B. Fan, G. Wang, and F. Wu, "Exploring local and overall ordinal information for robust feature description," *TPAMI*, vol. 38, no. 11, pp. 2198–2211, 2015.

[36] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "Lstm: A search space odyssey," *TNNLS*, vol. 28, no. 10, pp. 2222–2232, 2016.

[37] J. D. Robinson, C. Chuang, S. Sra, and S. Jegelka, "Contrastive learning with hard negative samples," in *ICLR*, 2021.

[38] "Cgraph," https://github.com/ChunelFeng/CGraph, 2021.

[39] C. Fu, C. Wang, and D. Cai, "High dimensional similarity search with satellite system graph: Efficiency, scalability, and unindexed query compatibility," *TPAMI*, 2021.

[40] W. Dong, M. Charikar, and K. Li, "Efficient k-nearest neighbor graph construction for generic similarity measures," in *WWW*, 2011, pp. 577–586.

[41] W. Li, Y. Zhang, Y. Sun, W. Wang, M. Li, W. Zhang, and X. Lin, "Approximate nearest neighbor search on high dimensional data - experiments, analyses, and improvement," *TKDE*, vol. 32, no. 8, pp. 1475–1488, 2020.

[42] W. Zhao, S. Tan, and P. Li, "SONG: approximate nearest neighbor search on GPU," in *ICDE*, 2020, pp. 1033–1044.

[43] J. Gao and C. Long, "High-dimensional approximate nearest neighbor search: with reliable and efficient distance comparison operations," *SIGMOD*, vol. 1, no. 2, pp. 137:1–137:27, 2023.

[44] P. Isola, J. J. Lim, and E. H. Adelson, "Discovering states and transformations in image collections," in *CVPR*, 2015, pp. 1383–1391.

[45] Y. Hou, E. Vig, M. Donoser, and L. Bazzani, "Learning attribute-driven disentangled representations for interactive fashion retrieval," in *ICCV*, 2021.

[46] A. Neculai, Y. Chen, and Z. Akata, "Probabilistic compositional embeddings for multimodal image retrieval," in *CVPR*, 2022, pp. 4547–4557.

[47] "Datasets for approximate nearest neighbor search," http://corpus-texmex.irisa.fr/, 2010.

[48] "Million song dataset benchmarks," http://www.ifs.tuwien.ac.at/mir/msd/download.html, 2023.

[49] "Uq video," https://github.com/Lsyhprum/WEAVESS/tree/dev/dataset, 2021.

[50] "Billion-scale approximate nearest neighbor search challenge: Neurips'21 competition track," https://big-ann-benchmarks.com/, 2021.

[51] S. Jayaram Subramanya, F. Devvrit, H. V. Simhadri, R. Krishnawamy, and R. Kadekodi, "Diskann: Fast accurate billion-point nearest neighbor search on a single node," in *NeurIPS*, vol. 32, 2019.

[52] J. A. V. Muñoz, M. A. Gonçalves, Z. Dias, and R. da Silva Torres, "Hierarchical clustering-based graphs for large scale approximate nearest neighbor search," *Pattern Recognition*, vol. 96, p. 106970, 2019.

[53] "Openai embeddings api," https://platform.openai.com/docs/guides/embeddings, 2023.

[54] "Hugging face embeddings api," https://huggingface.co/blog/getting-started-with-embeddings, 2023.

[55] J. Zhang, J. Du, and L. Dai, "A gru-based encoder-decoder approach with attention for online handwritten mathematical expression recognition," in *ICDAR*, vol. 1, 2017, pp. 902–907.

[56] C. Li, M. Zhang, D. G. Andersen, and Y. He, "Improving approximate nearest neighbor search through learned adaptive early termination," in *SIGMOD*, 2020, pp. 2539–2554.

[57] R. Guo, X. Luan, L. Xiang, X. Yan, X. Yi, J. Luo, Q. Cheng, W. Xu, J. Luo, F. Liu, Z. Cao, Y. Qiao, T. Wang, B. Tang, and C. Xie, "Manu: A cloud native vector database management system," *PVLDB*, vol. 15, no. 12, pp. 3548–3561, 2022.

[58] P. Zhang, B. Yao, C. Gao, B. Wu, X. He, F. Li, Y. Lu, C. Zhan, and F. Tang, "Learning-based query optimization for multi-probe approximate nearest neighbor search," *VLDBJ*, pp. 1–23, 2022.

[59] H. Jégou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *TPAMI*, vol. 33, no. 1, pp. 117–128, 2011.

[60] K. Lu, M. Kudo, C. Xiao, and Y. Ishikawa, "Hvs: hierarchical graph structure based on voronoi diagrams for solving approximate nearest neighbor search," *PVLDB*, vol. 15, no. 2, pp. 246–258, 2022.

[61] M. Li, Y.-G. Wang, P. Zhang, H. Wang, L. Fan, E. Li, and W. Wang, "Deep learning for approximate nearest neighbour search: A survey and future directions," *TKDE*, 2022.

[62] S. Dasgupta and Y. Freund, "Random projection trees and low dimensional manifolds," in *SOTC*, 2008, pp. 537–546.

[63] K. Lu, H. Wang, W. Wang, and M. Kudo, "VHP: approximate nearest neighbor search via virtual hypersphere partitioning," *PVLDB*, vol. 13, no. 9, pp. 1443–1455, 2020.

[64] M. Muja and D. G. Lowe, "Scalable nearest neighbor algorithms for high dimensional data," *TPAMI*, vol. 36, no. 11, pp. 2227–2240, 2014.

[65] R. Guo, P. Sun, E. Lindgren, Q. Geng, D. Simcha, F. Chern, and S. Kumar, "Accelerating large-scale inference with anisotropic vector quantization," in *ICML*, 2020, pp. 3887–3896.

[66] F. André, A. Kermarrec, and N. L. Scouarnec, "Cache locality is not enough: High-performance nearest neighbor search with product quantization fast scan," *PVLDB*, vol. 9, no. 4, pp. 288–299, 2015.

[67] Q. Huang, J. Feng, Y. Zhang, Q. Fang, and W. Ng, "Query-aware locality-sensitive hashing for approximate nearest neighbor search," *PVLDB*, vol. 9, no. 1, pp. 1–12, 2015.

[68] L. Gong, H. Wang, M. Ogihara, and J. Xu, "idec: Indexable distance estimating codes for approximate nearest neighbor search," *PVLDB*, vol. 13, no. 9, pp. 1483–1497, 2020.

[69] M. Li, Y. Zhang, Y. Sun, W. Wang, I. W. Tsang, and X. Lin, "I/O efficient approximate nearest neighbour search based on learned functions," in *ICDE*, 2020, pp. 289–300.

[70] "Benchmarks of approximate nearest neighbor libraries in python," https://github.com/erikbern/ann-benchmarks, 2021.

[71] I. Doshi, D. Das, A. Bhutani, R. Kumar, R. Bhatt, and N. Balasubramanian, "Lanns: A web-scale approximate nearest neighbor lookup system," *PVLDB*, vol. 15, no. 4, p. 850–858, 2022.

[72] Y. Zhu, L. Chen, Y. Gao, B. Zheng, and P. Wang, "Desire: An efficient dynamic cluster-based forest indexing for similarity search in multi-metric spaces," *PVLDB*, vol. 15, no. 10, pp. 2121–2133, 2022.

[73] M. Franzke, T. Emrich, A. Züfle, and M. Renz, "Indexing multi-metric data," in *ICDE*, 2016, pp. 1122–1133.

[74] "Billion-scale anns benchmarks," https://big-ann-benchmarks.com/, 2021.