APPENDIX

A ENCODERS USED IN OUR EXPERIMENTS

ResNet. ResNet is a type of deep neural network that uses residual blocks and skip connections to ease the training of very deep networks and avoid the problem of vanishing or exploding gradients. ResNet was proposed by researchers at Microsoft Research in 2015 [35] and won the ImageNet classification task with a 152-layer network. ResNet can also be used for other visual recognition tasks such as object detection and segmentation. We used ResNet17 and ResNet50 to encode the image modality in our experiments. They are variants of ResNet with different numbers of layers. ResNet17 has 17 layers, while ResNet50 has 50 layers. ResNet50 also uses a bottleneck design for its residual blocks, which reduces the number of parameters and speeds up the training process. Both ResNet17 and ResNet50 can be used as feature extractors for other tasks such as object detection or segmentation.

LSTM. LSTM stands for Long Short-Term Memory and it is a type of recurrent neural network (RNN) that can process sequential data such as speech or video [31]. LSTM has feedback connections and a special structure called a cell that can store and update information over long time intervals. LSTM also has three gates (input, output and forget) that regulate the flow of information into and out of the cell. LSTM can be used for various tasks such as speech recognition, machine translation, handwriting recognition, etc. LSTM can also be combined with convolutional neural networks (CNNs) to form a convolutional LSTM network, which can be used for tasks such as video prediction or object tracking. In our experiments, we used LSTM to encode the text modality.

Transformer. Transformer is a deep learning model that was introduced in 2017 for natural language processing tasks such as machine translation and text summarization [21]. Unlike recurrent neural networks, Transformer does not process sequential data in order, but rather uses a mechanism called attention to capture the dependencies between words or tokens. Transformer consists of two parts: an encoder and a decoder. The encoder takes an input sentence and converts it into a sequence of vectors called encodings. The decoder takes the encodings and generates an output sentence. Both the encoder and the decoder are composed of multiple layers, each containing a multi-head self-attention module and a feed-forward neural network module. Transformer also uses positional encodings to inject the information about the position of each word in the sentence. We also encoded the text modality with Transformer in our experiments.

Encoding. We used ordinal encoding [61] to encode the structured text description in our experiments. This technique converts categorical data into numerical data by assigning integer values to categories based on their rank or order. For example, if a feature has three categories: “low”, “medium”, and “high”, they can be encoded as 1, 2, and 3 respectively. Ordinal encoding is suitable for categorical features that have a natural ordering, such as grades, sizes, ratings, etc. We can reverse ordinal encoding by mapping the integer values to the original categories.

TIRG. TIRG (Text Image Residual Gating) is an approach for combining image and text features in the task of image retrieval [56]. It modifies the features of the query image using text, while ensuring that the resulting feature vector still “lives in” the same space as the target image. This is achieved through a gated residual connection, which allows for improved encoding and representation learning. We encoded image-text pairs as composition vectors with TIRG in our experiments.

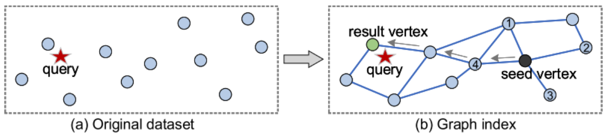


Figure 12: An example of indexing and search based on proximity graph [60].

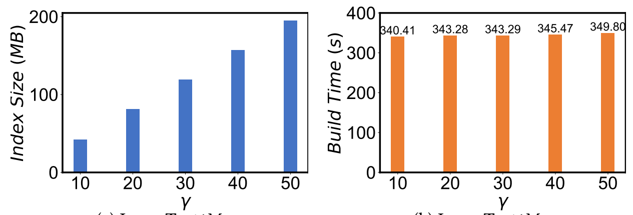
CLIP. CLIP (Contrastive Language-Image Pre-Training) is a neural network model developed by OpenAI that has achieved remarkable results in multi-modal zero-shot learning [53]. It is trained on a large dataset of image-text pairs scraped from the web and learns to associate images with their corresponding textual descriptions. CLIP has shown impressive generalization capabilities and has been successfully applied to various tasks such as fine-grained art classification, image generation, zero-shot video retrieval, event classification, and visual commonsense reasoning. We also used CLIP to encode image-text pairs as composition vectors in our experiments.

B PROXIMITY GRAPH-BASED INDEX ALGORITHM

Proximity graph-based algorithms are widely used for vector similarity search, especially in high-dimensional space. They offer a good balance between efficiency and accuracy by expressing neighbor relationship between vectors [28]. Many high-tech companies, such as Microsoft [40] and Alibaba [28], use them. To serve queries online, we need to build a proximity graph index offline on the dataset of feature vectors. The graph has vertices that correspond to the vector data points and edges that represent pairwise similarities or distances between vectors. Different algorithms, such as NSG [28] or KGraph [22], may use different graph construction methods. In a recent survey [60], we can find a thorough analysis of various proximity graph-based index algorithms, including their performance, strengths, and pitfalls. Fig. 12 shows an example of finding the nearest vertex to a query vector q using a proximity graph index. The process starts from a seed vertex (the black vertex), which can be random or fixed [60]. Then it visits its neighbors and computes their distance to q. It picks vertex 4 as the next visiting vertex because it is the closest to q among the seed's neighbors. This process repeats until it reaches the green vertex, which has no neighbors closer to q than itself. The search process depends on various factors, such as the seed acquisition strategy [49] and the routing technique [43].

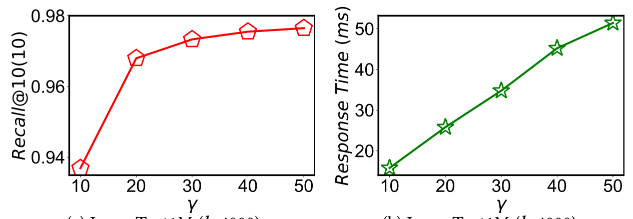
C PARAMETERS FOR FUSED INDEX

The fused index construction depends on two key parameters: maximum number of neighbors γ and maximum iterations ϵ. These parameters affect the index size, index build time, and search performance. Here, we evaluate some details about them.



(a) ImageText1M (b) ImageText1M

Figure 13: Index size and index build time under different values of γ.



(a) ImageText1M (l=4000) (b) ImageText1M (l=4000)

Figure 14: Search performance under different values of γ.

Table 10: Graph quality under different number of iterations.

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

Maximum number of neighbors γ. Fig. 13 shows that the index size and index build time increase with γ. This is because we need to handle more vertices in the *Initialization*, *Candidate acquisition*, and *Neighbor selection* components, which increases the complexity of index construction. Obviously, more neighbors also result in more index size. In Fig. 14, we keep other parameters unchanged (including the search parameters k and l) and report the search performance under different γ. We can see that the recall rate increases with γ. This is because we can visit more neighbors of a vertex during search for a large γ, which improves the search accuracy. However, when γ is very large, we need to compute more vector distances, which leads to low efficiency. As a result, we seek the best compromise between accuracy and efficiency by adjusting γ in practice. In our experiments, we set γ to 30 by default.

Maximum iterations ε. Graph quality is the mean ratio of γ neighbors of a vertex over the top-γ nearest neighbors based on the joint similarity [60]. Tab. 10 shows how graph quality changes with different number of iterations. The results show that the graph quality increases with ε and is close to 1 when ε is 3. Therefore, we set ε to 3 by default in all our experiments.

D PARAMETER FOR JOINT SEARCH

The result set size l (Algorithm 2) affects the accuracy and efficiency when searching. Here, we evaluate how the recall rate and response time change with l. As shown in Tab. 11, the recall rate and response time (per query) increase with l. This is because a larger l requires visiting more vertices and performing more vector calculations.

Table 11: Search performance under different values of l (γ = 30).

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

E DATASETS

CelebA. CelebFaces Attributes Dataset (CelebA) [47] is a large-scale face attributes dataset with more than 200K celebrity images. Each image has 40 attribute annotations, 5 landmark locations, and a face identity. The images cover large pose variations and background clutter. CelebA has large diversities, quantities, and rich annotations. The dataset can be used for various computer vision tasks, such as face attribute recognition, face recognition, face detection, landmark localization, and face editing & synthesis.

MIT-States. MIT-States [38] is a dataset consisting of approximately 60,000 images, each labeled with an object/noun label and a state/adjective label (such as "red tomato" or "new camera"). The dataset contains 245 nouns and 115 adjectives, with an average of only about 9 adjectives per noun. MIT-States is commonly used to evaluate both image retrieval and image classification tasks in the field of computer vision.

Shopping. The Shopping100k dataset [36] is a dataset containing 101,021 images consisting of pure clothing items extracted from different e-commerce providers for fashion studies. It was developed to address the limitations of existing datasets in fashion-related research, which often consist of posed images with occlusion problems. Each image in the dataset is represented with general and special attributes, where the special attributes are more suitable for attribute manipulation and thus conducting fashion searches. Shopping has various categories, such as T-shirt and bottoms.

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 [1]. Each vector represents an image and has a text modality added to it to form a multimodal dataset.

AudioText1M. AudioText1M is a semi-synthetic dataset that combines real-world audio with text. It contains a million contemporary popular music tracks with 420 dimensions of audio features and metadata [11]. Each vector represents an audio track and has a text modality added to it to form a multimodal dataset.

VideoText1M. VideoText1M is a semi-synthetic dataset that combines real-world video with text. It extracts 256 dimensions of local features from some keyframes of each video [8]. Each vector represents a video and has a text modality added to it to form a multimodal dataset.

ImageText16M. ImageText16M is a semi-synthetic dataset that combines real-world images with text. It consists of 16 million data points with 96 dimensions of deep neural codes of natural images. These codes are obtained from the activations of a convolutional neural network [3]. Each vector represents an image and has a text modality added to it to form a multimodal dataset.

Table 12: 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 13: 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 14: 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 15: Output weights of module for CelebA+ dataset.

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

Table 16: 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

Table 17: 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

F WEIGHTS SETTING

In MUST, we use a vector weight learning module to capture the importance of different modalities. In Tab. 12-16, we show the specific weights used for building index and query processing on different datasets and encoders.

G SEARCH ACCURACY ONLY USING AUXILIARY MODALITY

Tab. 17 shows the search accuracy using only the auxiliary modality on three real-world datasets. In general, they perform worse than the methods that combine multiple modalities. However, in some cases, they are better than JE, which fuses the features of all modality inputs into a joint embedding. A possible explanation is that JE introduces large encoder error while the auxiliary modality can describe an object accurately in some datasets. This further verifies the necessity of using multiple vectors from different encoders to represent an object.

Table 18: 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%)**

H SEARCH ACCURACY ON SHOPPING (BOTTOMS)

Tab. 18 shows the search accuracy for the "bottoms" category of the Shopping dataset. Different categories in the Shopping dataset share the same output weights. This demonstrates the generalization ability of the vector weight learning module.

I RECALL EXAMPLES ON REAL-WORLD DATASETS

Given a query input with *a fresh cheese image* and *a text description "change state to moldy"*, we show its top-10 search results with different frameworks in Fig. 15. The results show that MUST recalls all target objects, while MR and JE recall only a few target objects and many results match only the text description.

Given a query input with *a male face image* and *a text description "change state to bags under eyes, high cheekbones, mouth slightly open, and smiling"*, we show its top-10 search results with different frameworks in Fig. 16. Note that we also require that the target face and the query input face belong to the same identity, besides matching the query input face and text description. From the results, we can see MUST gets more target faces.

Given a query input with *a female face image* and *a text description "change state to arched eyebrows and pointy nose"*, we show its top-10 search results with different frameworks in Fig. 17. Besides matching the query input face and text description, we also require that the target face and the query input face belong to the same identity. This requirement makes it harder to get the target face. Even so, the results show that MUST gets more target faces.

Given a query input with *a T-shirt image* and *a text description "replace gray color with white color and replace sweat fabric with jersey fabric"*, we show its top-10 search results with different frameworks in Fig. 18. From the results, we can see MUST gets more target T-shirts.

Given a query input with *a T-shirt image* and *a text description "replace sweat fabric with jersey fabric and replace striped pattern with print pattern"*, we show its top-10 search results with different frameworks in Fig. 19. From the results, we can see MUST gets more target T-shirts.

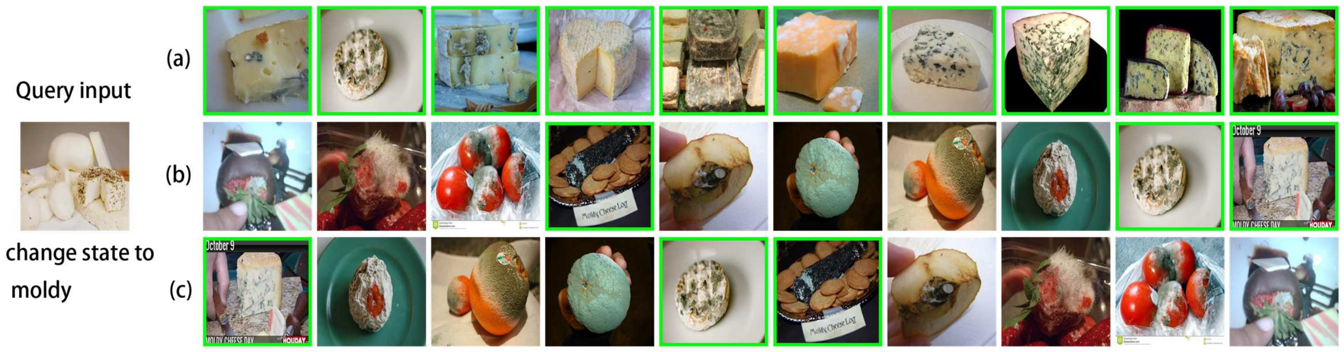


Figure 15: Some recall examples with different frameworks and optimal encoders on MIT-States dataset. (a), (b), and (c) are the top-10 results of MUST, MR, and JE, respectively. The green box marks the target objects.

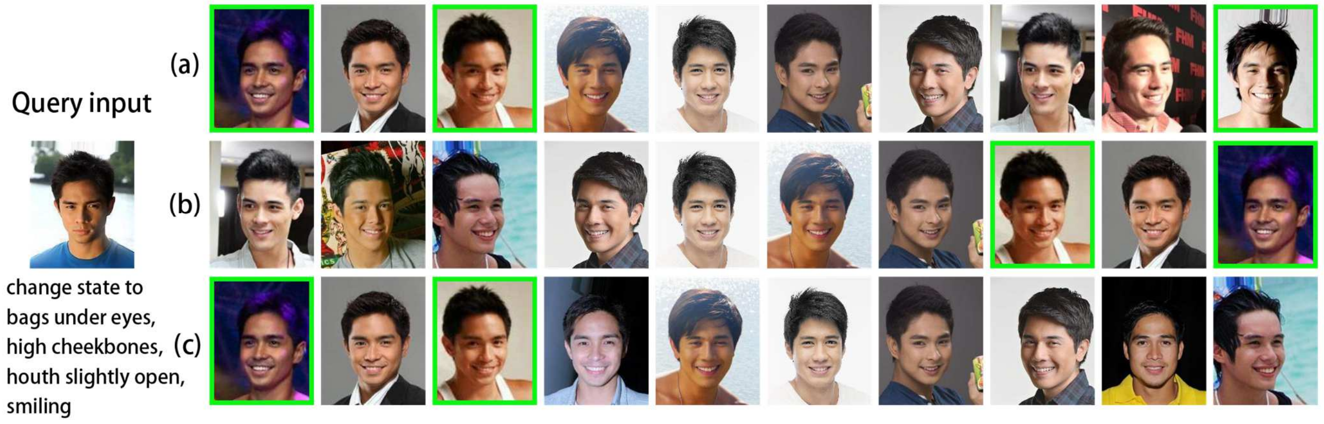


Figure 16: Some recall examples with different frameworks and optimal encoders on CelebA dataset. (a), (b), and (c) are the top-10 results of MUST, MR, and JE, respectively. The green box marks the target objects.

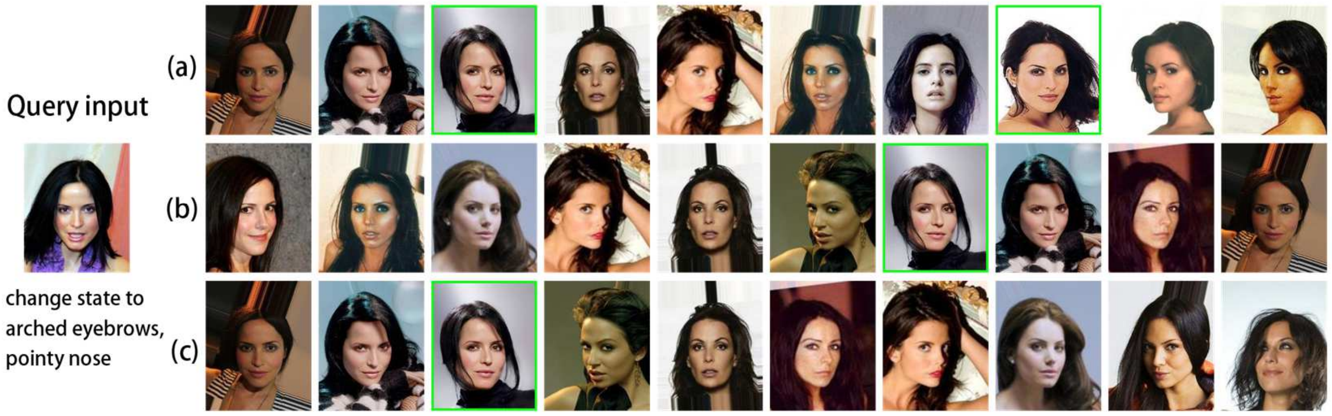


Figure 17: Some recall examples with different frameworks and optimal encoders on CelebA dataset. (a), (b), and (c) are the top-10 results of MUST, MR, and JE, respectively. The green box marks the target objects.

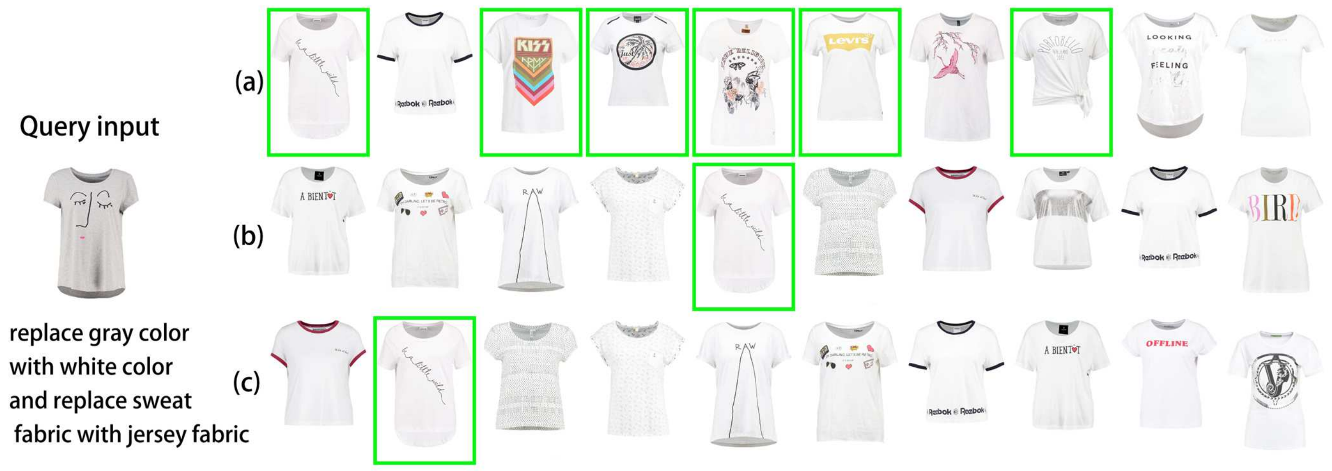

Query input
replace gray color with white color and replace sweat fabric with jersey fabric
(a)
(b)
(c)

Figure 18: Some recall examples with different frameworks and 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.

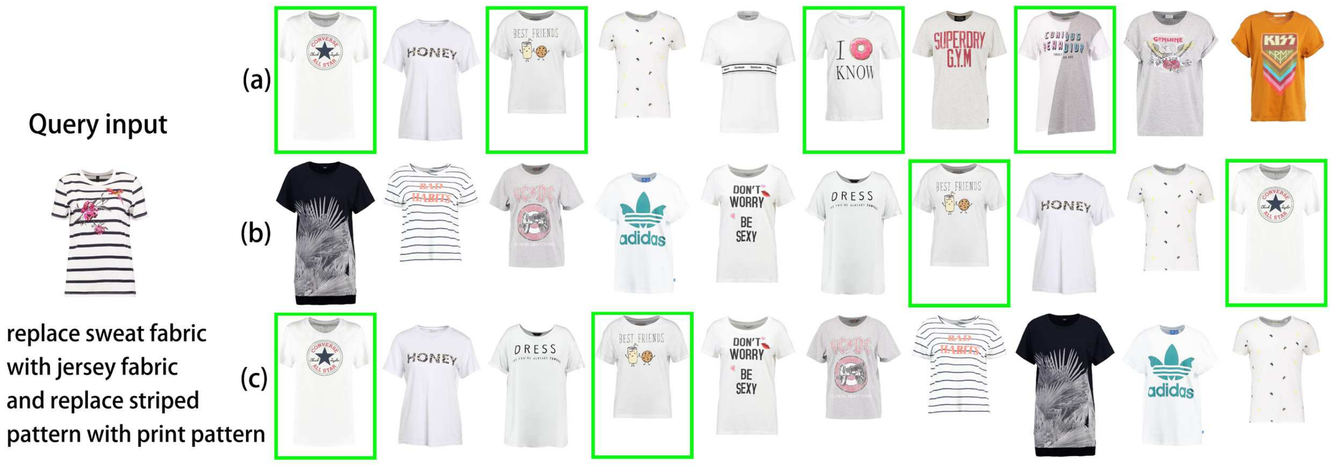

Query input
replace sweat fabric with jersey fabric and replace striped pattern with print pattern
(a)
(b)
(c)

Figure 19: Some recall examples with different frameworks and 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.