


Article

An Efficient Cross-Modal Privacy-Preserving Image–Text Retrieval Scheme

Kejun Zhang^{1,2}, Shaofei Xu^{1,*} , Yutuo Song², Yuwei Xu³, Pengcheng Li², Xiang Yang¹, Bing Zou¹ and Wenbin Wang^{1,*}¹ Beijing Electronic Science and Technology Institute, Beijing 100070, China² School of Computer Science and Technology, Xidian University, Xi'an 710071, China³ School of Automation and Information Engineering, Xi'an University of Technology, Xi'an 710048, China

* Correspondence: 20222926@mail.besti.edu.cn (S.X.); 20212923@mail.besti.edu.cn (W.W.)

Abstract: Preserving the privacy of the ever-increasing multimedia data on the cloud while providing accurate and fast retrieval services has become a hot topic in information security. However, existing relevant schemes still have significant room for improvement in accuracy and speed. Therefore, this paper proposes a privacy-preserving image–text retrieval scheme called Pitr. To enhance model performance with minimal parameter training, we freeze all parameters of a multimodal pre-trained model and incorporate trainable modules along with either a general adapter or a specialized adapter, which are used to enhance the model's ability to perform zero-shot image classification and cross-modal retrieval in general or specialized datasets, respectively. To preserve the privacy of outsourced data on the cloud and the privacy of the user's retrieval process, we employ asymmetric scalar-product-preserving encryption technology suitable for inner product calculation, and we employ distributed index storage technology and construct a two-level security model. We construct a hierarchical index structure to speed up query matching among massive high-dimensional index vectors. Experimental results demonstrate that our scheme can provide users with secure, accurate, fast cross-modal retrieval service while preserving data privacy.

Keywords: privacy-preserving; searchable encryption; image–text retrieval; cross-modal retrieval

Citation: Zhang, K.; Xu, S.; Song, Y.; Xu, Y.; Li, P.; Yang, X.; Zou, B.; Wang, W. An Efficient Cross-Modal Privacy-Preserving Image–Text Retrieval Scheme. *Symmetry* **2024**, *16*, 1084. <https://doi.org/10.3390/sym16081084>

Academic Editor: Sergei D. Odintsov

Received: 1 July 2024

Revised: 7 August 2024

Accepted: 9 August 2024

Published: 21 August 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

With the rapid development of the Internet and information technology, personal multimedia data with pictures and text as the main types have shown explosive growth, and with them have come an increasing demand for storage space. Therefore, more and more users store private multimedia data in third-party cloud servers to save local storage space. How to preserve data privacy while providing users with accurate and fast cross-modal data retrieval services has become a hot issue that requires urgent research.

However, current privacy-preserving data retrieval solutions face many issues. First, most solutions are designed only for single-modal data like text or images [? ? ? ? ? ? ? ? ? ? ? ?]. They cannot meet the needs for storing and retrieving multimedia data, which greatly limits the users' scope. For instance, when users wish to retrieve text and images of apples stored in the cloud using a text query "apple", the current unimodal retrieval methods based on text or images alone are inadequate. Our proposed cross-modal retrieval scheme can address this issue effectively. Cross-modal retrieval requires solving the semantic alignment of multimodal data. This means accurately mapping the features of multimodal data to a common semantic space [?]. Second, existing solutions can only meet general retrieval needs. When the data involve specialized knowledge, the accuracy of retrieval decreases. If training is conducted on data from numerous specialized domains, the limited expressive capacity of the parameters will be a constraint. Third, current solutions mainly use traditional indexing structures for retrieval, such as forward index structure [?], inverted index structure [?], tree-based index structure [?], and hash

index structure [?]. However, these structures become inefficient when searching among large amounts of high-dimensional vectors.

To address the issues mentioned above, this paper proposes an efficient privacy-preserving cross-modal image–text retrieval scheme called PITR. The main contributions of this paper are as follows:

- **General adapter structure:** We propose a general adapter structure that introduces a set of trainable pseudo-prompt vectors and an encoder to map these vectors to a specific space on the text input end of a frozen multimodal pre-trained model. The vectors generate appropriate prompts for different tasks. Our general adapter only needs to train about 5.23% of the parameters to improve performance in image–text retrieval and zero-shot image classification tasks in general domain datasets.
- **Specialized adapter structure:** We propose a specialized adapter structure that incorporates learnable rescaling vectors at specific positions in the multi-head attention and feed-forward layers of the transformer networks on both the text and visual ends of a frozen multimodal pre-trained model in order to perform fine-grained scaling of computation results. The original model equipped with this structure only needs to train about 0.017% of the parameters to improve performance in image–text retrieval and zero-shot image classification tasks in specialized domain datasets.
- **Diverse retrieval interfaces:** We provide users with both general domain and specialized domain retrieval interfaces. For users’ general retrieval needs, only the trained general adapter needs to be installed on the base model. For data from different specialized domains, it is only necessary to train and install lightweight specialized adapter modules corresponding to those domains. This approach offers users diverse and flexible cross-modal retrieval services.
- **Efficient cross-modal retrieval structure:** We design an efficient cross-modal retrieval scheme. The hierarchical navigable small world (HNSW) algorithm is used instead of traditional indexing structures. This improves the efficiency of approximate nearest neighbor (ANN) searches for users’ query trapdoors among a large number of high-dimensional encrypted feature indexing vectors.
- **Comprehensive privacy protection mechanism:** We develop a comprehensive privacy protection mechanism with two levels of security modes that are adaptable to different user tasks. The level I security mode is based on our proposed index distributed storage technology, and the level II security mode is based on asymmetric scalar-product-preserving encryption (ASPE) technology suitable for inner product calculation.

The remainder of this paper is organized as follows. Section 2 reviews the current state of research related to the issues addressed in this study, including privacy-preserving image retrieval schemes and cross-modal retrieval schemes. Section 3 presents the main background knowledge and techniques required for designing the PITR scheme. Section 4 introduces the system framework of PITR, including the system components and the overall operation process. Section 5 details the main methods used in PITR and the specific implementation details of its three main modules (feature extraction model, encryption model, and retrieval model). Section 6 evaluates the performance of PITR in zero-shot image classification tasks and cross-modal image–text retrieval tasks. Finally, Section 7 concludes the paper by summarizing the overall value and significance of the proposed scheme and discussing directions for future research.

2. Related Work

2.1. Privacy-Preserving Image Retrieval Schemes

Existing privacy-preserving image retrieval schemes focus on image-to-image searches. Among them, privacy-preserving content-based image retrieval (PPCBIR) is the most widely studied. These solutions can be categorized into three types:

1. Images are encrypted and then uploaded to the cloud for feature extraction [14], and we use the extracted features to build an unsupervised or supervised retrieval model. For this approach, the task of feature extraction from encrypted images can be outsourced to the cloud. Despite encryption preserving the privacy of images on the cloud, accurately and comprehensively extracting dense semantic features from encrypted images is challenging [15]. Most existing solutions can only extract coarse-grained features and use simple retrieval models [16] such as k-means and bag-of-words models. However, these models struggle to learn the complex nonlinear features in images. Additionally, a multi-level feature extraction method for images should be compatible with the encryption algorithm used. Therefore, existing schemes generally choose to extract weak features that differ from but are somewhat related to the original images. They also need to design suitable encryption algorithms and multi-level weak feature extraction algorithms that are compatible with the encryption algorithms [17]. However, while these approaches can ensure a certain degree of privacy and offload the feature extraction task to the cloud, using manually designed weak features instead of the complex fine-grained semantic features in the original images significantly reduces retrieval accuracy.
2. Homomorphic encryption is used to modify the retrieval model on the cloud to achieve secure inference [18]. Although these schemes can accomplish secure image retrieval tasks, they have inherent limitations. Firstly, homomorphic encryption algorithms only support integer-type data, while the data and model parameters in machine learning are typically floating-point numbers. Secondly, fully homomorphic encryption (FHE) does not support nonlinear operations and can only approximate results using approximate functions [19]. However, deep learning models that accurately capture data features involve numerous nonlinear operations, leading to a loss of computational accuracy in the original models.
3. Features are extracted from original images before encryption [20]. Then, the encrypted images and their encrypted features are sent to the cloud. For retrieval, the user needs to construct a query trapdoor and send it to the cloud; then, the cloud calculates the similarity between the query trapdoor and the encrypted indexes and returns the most relevant results. This type of scheme transfers the feature extraction work to the user, increasing the user's workload. However, it allows the use of larger models to extract complex dense semantic features from data, enabling cross-modal retrieval and ensuring maximum retrieval accuracy.

2.2. Privacy-Preserving Cross-Modal Image–Text Retrieval Schemes

Currently, there is limited research on privacy-preserving cross-modal retrieval. Ref. [21] proposed the first privacy-preserving cross-modal retrieval system. This method performs cross-modal retrieval on special hardware: the Intel SGX. However, original data may be leaked if the system is attacked. Ref. [22] is the first to study privacy-preserving cross-modal retrieval using deep hashing and adversarial learning. The authors built a two-branch GAN hashing model to extract features, which improved retrieval accuracy. However, the feature extraction and index construction process is computationally complex. Ref. [23] proposed a privacy-preserving cross-media retrieval scheme, PPCMR, for encrypted data in the cloud. They constructed a two-branch feature extraction model based on convolutional neural networks (CNNs) to learn the semantic gap between different modalities of encrypted data. However, since it requires retraining the two-branch feature extraction network, the retrieval accuracy is not high.

3. Preliminary Knowledge

3.1. Contrastive Language-Image Pre-Training (CLIP)

CLIP [24] is a dual-stream structure image–text pre-trained large model that uses 400 million image–text pairs collected from the internet and is trained using a contrastive

learning method. The model performs very well in zero-shot image classification and image–text matching tasks.

In CLIP’s training process, shown in Figure 1, the text and visual feature extractors encode the text and images into embeddings of the same dimension. Then, the embeddings are used to calculate their similarity. In contrastive learning, the goal is to maximize the diagonal values of the similarity matrix, which represents paired image–text pairs, and to minimize the off-diagonal values.

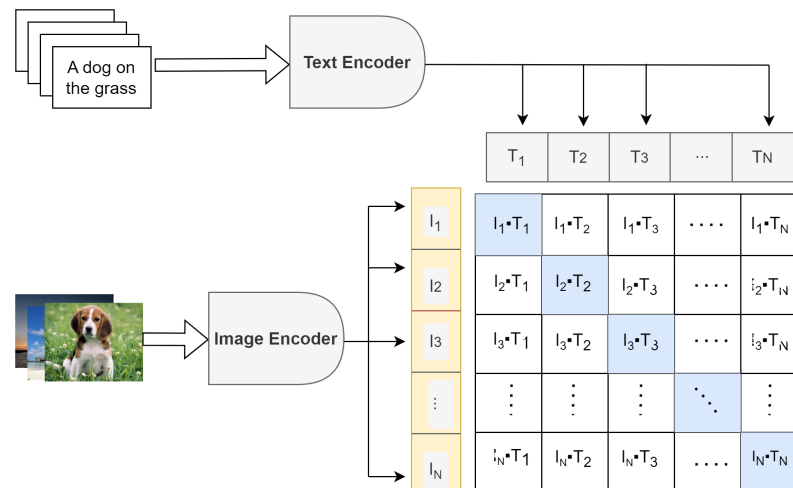


Figure 1. Contrastive learning training process of CLIP model.

Figure 2 illustrates the process of using CLIP for zero-shot image classification. The image labels are appended to pre-designed prompt templates to form prompt sentences, such as “A photo of a {class}.” These prompt sentences are then processed by the text encoder to obtain the global semantic feature embedding. On the visual side, the image is sent to the image encoder to obtain the global feature embedding of the image with the same dimensions as the prompt embedding. Finally, the dot product is computed between the prompt embedding and the image embedding. The result represents the degree to which the image matches each label, and the label corresponding to the highest value is the one predicted by CLIP.

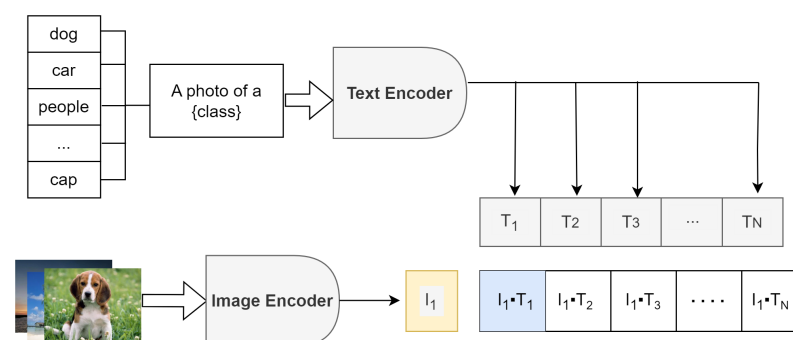


Figure 2. Zero-shot image classification process of CLIP model.

3.2. Hierarchical Navigable Small World Graphs (HNSW)

In 2018, ref. [?] proposed the HNSW algorithm for vector nearest neighbor searches; it is currently the best-performing ANN algorithm overall. The HNSW algorithm constructs its index by incrementally inserting elements into a multi-layer graph structure. Each element is inserted starting from the top layer, where it is connected to a subset of existing elements based on proximity. As the insertion progresses to lower layers, the connections become denser, ensuring that each element is linked to its nearest neighbors. This hierar-

chical approach allows the algorithm to maintain a balance between search efficiency and accuracy. The sparsity of connections in higher layers facilitates rapid traversal, while the denser connections in lower layers ensure precise neighbor identification.

When performing a search on the constructed HNSW index, the algorithm begins at the top layer with a randomly selected entry point. It then traverses the graph by iteratively moving to the closest neighbor of the current node, gradually descending through the layers. At each layer, the search process refines the candidate set of nearest neighbors by exploring nodes within a defined radius. This hierarchical search strategy enables the algorithm to efficiently navigate the graph, leveraging the sparse connections in higher layers for rapid exploration and the dense connections in lower layers for accurate neighbor retrieval. Ultimately, this approach achieves a balance between speed and precision in nearest neighbor searches.

In contrast, the index structures used in traditional searchable encryption schemes (including forward indexes, inverted indexes, cluster indexes, and hash indexes) are designed based on keyword matching. They require the precise extraction of keywords from each document and the construction of a large keyword set, which becomes inefficient and unsuitable for semantic retrieval based on feature vector representation. Moreover, in some schemes that implement semantic searches using feature vectors, refs. [??] perform pairwise matching computations directly on all vectors. This method is inefficient for handling large-scale datasets because the computational load increases rapidly with the data size, leading to significant increases in search time and resource consumption.

3.3. Singular Value Decomposition (SVD)

The singular value decomposition (SVD) algorithm [?] is one of the most important matrix decomposition methods in machine learning. It decomposes the original matrix into three matrices: an orthogonal matrix, a diagonal matrix, and another orthogonal matrix. As shown in Figure 3, any matrix $A(m \times n)$ can be decomposed as: $A = U\Sigma V^T$. Here, $U(m \times m)$ is an orthogonal matrix formed by the eigenvectors of AA^T and is called the left singular matrix. $\Sigma(m \times n)$ is a diagonal matrix, with diagonal elements being the square roots of the eigenvalues of AA^T or $A^T A$, which are arranged in descending order; this matrix is called the singular value matrix. $V(n \times n)$ is an orthogonal matrix formed by the eigenvectors of $A^T A$ and is called the right singular matrix.

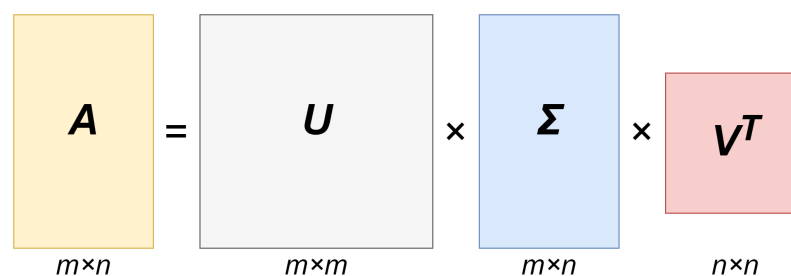


Figure 3. SVD matrix decomposition method.

It is worth noting that not all matrices can be decomposed using SVD. However, for any real-valued matrix, there exists a closest matrix that can be decomposed using SVD; this is known as the optimal approximation theorem. Additionally, the singular values on the diagonal of the singular value matrix σ are arranged in descending order, and these singular values decrease rapidly [?]. Therefore, we can approximate the original matrix using the largest k singular values and their corresponding left and right singular matrices. This approach helps save local storage space but also leads to a loss of the semantic information represented by the matrix. This is shown in Equation (1):

$$A_{m \times n} = U_{m \times m} \Sigma_{m \times n} V_{n \times n}^T \approx U_{m \times k} \Sigma_{k \times k} V_{k \times n}^T \quad (1)$$

3.4. Asymmetric Scalar-Product-Preserving Encryption (ASPE)

Wong et al. [?], based on their research on k-nearest neighbor (KNN) computation [?], proposed the ASPE algorithm, which supports KNN computation on encrypted data. The basic idea of ASPE is that data owners first encrypt all tuples using an encryption method Enc_A before uploading the data tuples to the server. The encrypted data are then sent to the cloud server. Queries are encrypted using another encryption method Enc_B , and Enc_A and Enc_B are different. These two encryption methods ensure that the product of the encrypted query tuple and the encrypted database tuples is preserved: that is, $pq = Enc_A(p)Enc_B(q)$. Additionally, it is required that for any two tuples e_i and e_j in the database, $e_i e_j \neq Enc_A(e_i)Enc_A(e_j)$. The original ASPE technique is based on Euclidean distances for computing the distances between vectors. To adapt this technique for inner product similarity computation between vectors, ref. [?] improved the algorithm in their proposed MRSE scheme. Inspired by their method, PITR will use the improved ASPE algorithm to encrypt data feature embeddings.

4. System Framework

4.1. System Composition

The operation of PITR is realized through the connection and interaction between different entities, which include:

Data Owner (DO): Fully trusted, can access all plaintext data in the system, and has unique key management permissions.

Cloud Server (CS): “Honest but curious”, meaning it will honestly execute user query requests but will also try to peek into users’ privacy. It hosts the retrieval model used for index construction and searches.

Private Server (PS): Fully trusted and attached to *DO*. It can share all keys with *DO* and hosts the feature extraction model used for extracting features from original images and text data.

Data User (DU): Needs prior authorization from *DO* to query, sends text or image type queries to *PS*, and receives the related query result from *PS*.

4.2. System Operation

Our overall system framework is shown in Figure 4. The complete process during system operation is as follows:

- (1) *DO* sends the original data to *PS* and the encrypted data to *CS*.
- (2) *PS* extracts features from the original data and uploads the encrypted feature embeddings set to *CS*.
- (3) When *DO* or *DU* need to perform a search, they send a query to *PS*; then, *PS* extracts feature embeddings from the query, encrypts them to form a query trapdoor, and sends them to *CS*.
- (4) *CS* uses the encrypted embeddings uploaded by *PS* to construct the index structure. Upon receiving a query trapdoor from *PS*, *CS* runs the retrieval model and returns the most relevant query results to *PS*.
- (5) *PS* decrypts the received encrypted query results and returns the decrypted query results to *DO* or *DU*.

In the aforementioned process, the content transmitted between *DO*, *PS*, and *CS* is encrypted. However, the communication between *DO* and *PS*, as well as between *PS* and *DU*, is not encrypted. Therefore, to further enhance the security of the communication process, taking the interaction between *PS* and *DU* as an example, both parties need to generate their respective public and private keys and exchange public keys before communication, with identity authentication facilitated by a certificate authority. Simultaneously, *PS* generates a symmetric key K for subsequent encrypted communication and encrypts K using *DU*'s public key before sending it to *DU*. *DU* decrypts it with its private key to obtain the symmetric key K . Subsequently, *PS* and *DU* use symmetric encryption for secure

communication. This process is outlined briefly, and the specific encryption algorithms and communication techniques are not detailed further.

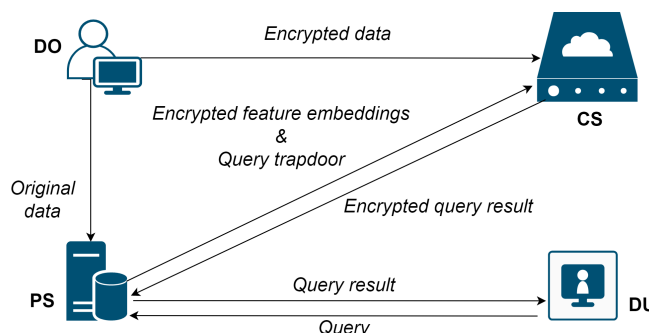


Figure 4. The system framework of PITR.

5. PITR: Privacy-Preserving Image–Text Retrieval Scheme

This section will introduce the implementation details of PITR. Our PITR scheme consists of three parts: the feature extraction model deployed on *PS*, the encryption model, and the secure retrieval model deployed on *CS*. This section will provide a detailed explanation of them.

5.1. Feature Extraction Model

The feature extraction model of PITR can be divided into a general domain feature extraction model and a specialized domain feature extraction model.

5.1.1. General Domain Feature Extraction Model

In the general domain data, our base model for feature extraction is CN-CLIP, as proposed in [?]. CN-CLIP is a multimodal pre-trained large model that is based on CLIP and is trained on a large-scale Chinese dataset. Inspired by the method of [? ?], we designed new training methods to further enhance the base model’s feature extraction capability.

It is worth noting that in zero-shot image classification tasks, both CLIP and CN-CLIP use manually designed prompt templates. However, these templates are very unstable. For example, changing “A photo of a {class}” to “This photo belongs to {class}” can lead to significantly different results, even though the semantic meaning seems unchanged from a human perspective. Therefore, a better choice is to replace manually designed templates with flexible, trainable templates. These templates consist of several learnable vectors. Although these cannot be translated into human-readable language, they overcome the limitations of traditional textual language in expressing semantic information. This enhances the model’s ability to capture and represent the semantics of the text.

Specifically, as shown in Figure 5, during the zero-shot image classification phase, we add a series of learnable vectors called pseudo-prompts and a prompt encoder to map them into a specific semantic space. The structure of the prompt encoder can be an LSTM [?] or an MLP network [?]. This training method enhances the model’s performance in zero-shot image classification tasks on general domain data. However, for image–text retrieval tasks in the general domain, the text is no longer simple words but longer sentences. Therefore, it is necessary to design a new training method to improve the general adapter’s ability to extract semantic features from longer texts like image captions. To achieve this, we switch from the previous multi-label learning method to the contrastive learning method and increase the number of pseudo-prompts while keeping the rest of the process unchanged. The specific training process is shown in Figure 6. It is important to note that during the training process of the general adapter, only the parameters of the pseudo-prompts and the prompt encoder need to be trained, while the rest of the network remains frozen. This accounts for only 5.23% of the total model parameters. Compared to fully fine-tuning 100% of the parameters, our approach significantly saves training time and computational resources.

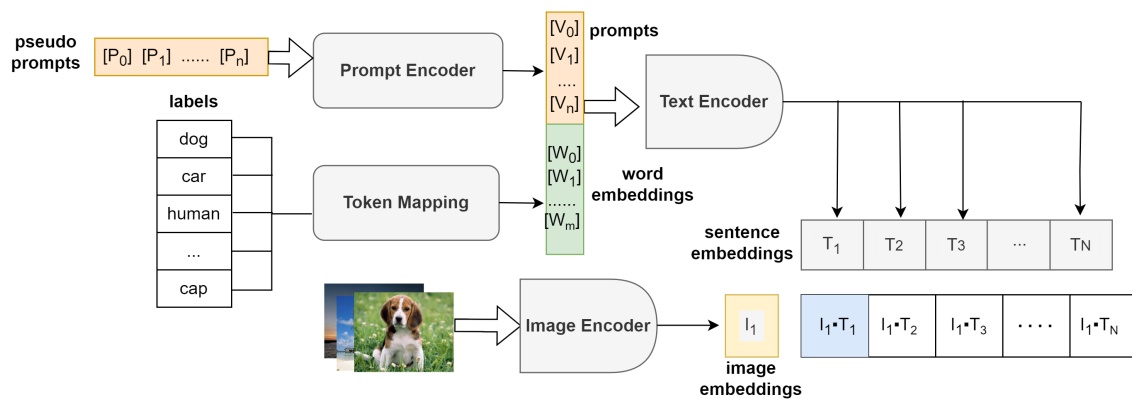


Figure 5. The zero-shot image classification task of PITR. **Step 1:** n pseudo-prompts ($[P_0], [P_1], \dots [P_n]$) are sent into the prompt encoder, which maps them to template embeddings (prompts: $[V_0], [V_1], \dots [V_n]$). **Step 2:** The image labels are sent to the token mapping module. This module tokenizes the labels, adds special tokens, performs token embedding mapping, and adds segment embeddings and position embeddings to get the word embeddings: $[W_0], [W_1], \dots [W_m]$. **Step 3:** The prompts and word embeddings are concatenated. The position of concatenation is flexible; prompts can be inserted at the beginning, middle, or end of the word embeddings. **Step 4:** The concatenated result is sent to the text encoder to get the global semantic feature embeddings of the entire sentence (sentence embeddings). These are then compared with the global semantic feature embeddings of the image produced by the image encoder using cosine similarity. The pseudo-prompts and the weights of the prompt encoder are optimized using the multi-label learning method.

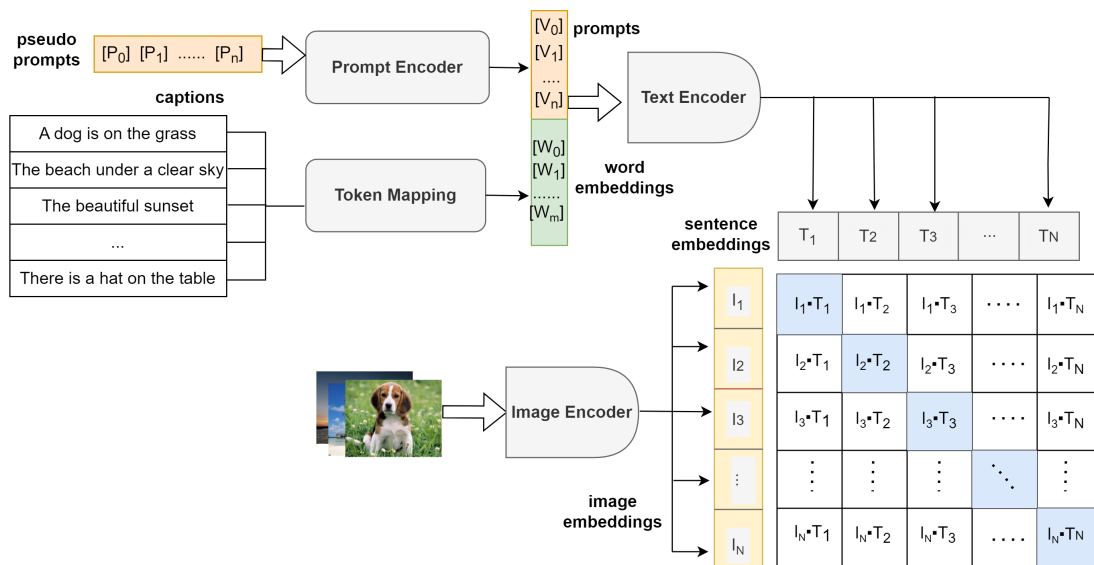


Figure 6. Similar to the training process of PITR in zero-shot image classification tasks, during the training process of PITR’s image–text matching task, pseudo-prompts and the prompt encoder generate prompts suitable for the task. These prompts can more accurately help determine whether an image and text are matched.

5.1.2. Specialized Domain Feature Extraction Model

When extracting features from specialized domain data, the base model remains unchanged. Inspired by [?], we add three sets of learnable rescaling vectors $l_k, l_v,$ and l_{ff} (collectively referred to as the specialized adapter), as shown in Figure 7. If the output at a particular position needs to be amplified, a positive learnable vector is added. Conversely, if it needs to be suppressed, a negative learnable vector is added, and the degree of amplification or suppression is determined by the vector’s value. Specifically, as shown in Equation (2), l_k and l_v are injected at the intermediate output of the key and value

sub-layers, respectively, where \odot represents element-wise multiplication, X represents a series of word vectors input to the self-attention layer, d_k represents the second dimension of W_K , and W_K , W_Q , and W_V represent the weights of the query, key, and value sub-layers, respectively. In the feed-forward layer of each transformer layer, as shown in Equation (3), l_{ff} is injected after the output of the nonlinear function. Here, x represents the input to the feed-forward layer. Additionally, it is important to note that l_{ff} injected into the feed-forward layer of each transformer layer, as well as l_k and l_v in each head of the multi-head self-attention layer, need to be independently trained. Assuming the number of layers in the transformer is N and the number of heads in each multi-head attention layer is H , the total number of parameters that need to be trained is $N(\text{Dim}_{l_{ff}} + H(\text{Dim}_{l_k} + \text{Dim}_{l_v}))$.

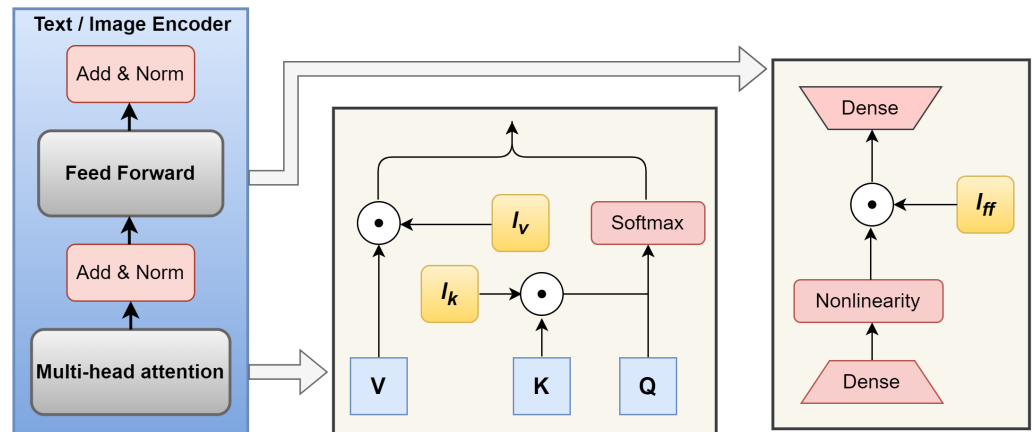


Figure 7. The learnable rescaling vectors l_k and l_v are injected into each head of the multi-head attention layer in every transformer block of both the text encoder and the image encoder, specifically at the intermediate output positions of the key and value sub-layers. Another learnable vector l_{ff} is injected after the output of the nonlinear function in the feed-forward network of each transformer layer. These learnable vectors can flexibly suppress or amplify the outputs of the key and value sub-layers in each self-attention layer and the outputs of the feed-forward network layer depending on the task.

$$\text{Softmax}\left(\frac{XW_Q[l_k \odot (XW_K)^T]}{\sqrt{d_k}}\right)(l_v \odot XW_V) \quad (2)$$

$$W_2(l_{ff} \odot \text{Relu}(xW_1 + b_1)) + b_2 \quad (3)$$

By using the specialized adapter to fine-tune the output results at different positions, the model can flexibly select the appropriate specialized adapter for training and inference based on the specialized knowledge in different domains.

Notably, before training, we freeze all parameters in the original model and only optimize the parameters of the learnable rescaling vectors, which account for a very small proportion of the model's total parameters, typically around 0.017%. During training, we use image–text datasets from different specialized domains to train the model and generate different specialized adapters. When performing downstream tasks on datasets from different specialized domains, we only need to load the corresponding specialized adapter. This enables the model to flexibly apply the specialized knowledge it has learned to complete various downstream tasks.

5.1.3. Dimensionality Reduction of the Extracted Embedding Vectors

The embedding vectors extracted by the feature extraction model are of very high dimensions. These embedding vectors will be subsequently used to construct HNSW indexes and compute distances with query vectors. If the computational resources of the cloud server are limited, dimensionality reduction of these high-dimensional embedding vectors can be performed first to reduce time and computational resource consumption.

However, it is important to note that dimensionality reduction will diminish the accurate representation of global semantic information in the original data for these embedding vectors, resulting in a loss of retrieval accuracy.

A common method for dimensionality reduction is the PCA algorithm [?], which can also be implemented using the SVD algorithm. This is because the eigenvectors of the covariance matrix required for PCA correspond to the right singular matrix obtained from SVD decomposition, and the eigenvalues correspond to the squares of the singular values [?]. The specific process of dimensionality reduction is as follows:

Assume that the feature embedding matrix of images and text is E , with dimensions $m \times n$. Here, m represents the total number of texts and images, and n represents the dimension of the feature embeddings.

(1) Using the SVD algorithm to decompose E , we obtain $E_{m \times n} = U_{m \times m} \Sigma_{m \times n} V_{n \times n}^T$.

(2) To calculate the mean value of each dimension of E , let $mean_j$ represent the mean value of the j -th column of E . If e_{ij} is the element in the i -th row and j -th column of E , then $mean_j = \frac{1}{m} \sum_{i=1}^m e_{ij}$.

(3) By subtracting the mean value of each corresponding column from every element in E , we obtain the centered matrix $E_{centered}$. For any element \hat{e}_{ij} in $E_{centered}$, we have $\hat{e}_{ij} = e_{ij} - mean_j$.

(4) Assuming the reduced dimension is k , we select the first k columns from V to form $V_{n \times k}$. Finally, we obtain the feature embedding matrix after PCA dimensionality reduction as $E_{PCA(m \times k)} = E_{centered(m \times n)} \times V_{(n \times k)}$.

5.2. Encryption Model

5.2.1. Symbol Definition and Description

The definitions of the main symbols used in our scheme are shown in Table 1. All sentences in each document of the text document collection D together form the text sentence collection S . We assume the total number of text documents is p , the total number of sentences in all documents is h , the total number of images is k , and the total number of images and text sentences is m .

Table 1. Symbol definitions.

Symbol	Description	Symbol	Description
$D = \{d_1, d_2, \dots, d_p\}$	Document set	$E_t = \{e_{t_1}, e_{t_2}, \dots, e_{t_n}\}$	Sentence embedding set
$S = \{s_1, s_2, \dots, s_h\}$	Sentence set	$E_I = \{e_{i_1}, e_{i_2}, \dots, e_{i_k}\}$	Image embedding set
$I = \{i_1, i_2, \dots, i_k\}$	Image set	$E = \{e_1, e_2, \dots, e_m\}$	Image–text embedding set
$\tilde{D} = \{\tilde{d}_1, \tilde{d}_2, \dots, \tilde{d}_p\}$	Encrypted document set	$\tilde{E} = \{\tilde{e}_1, \tilde{e}_2, \dots, \tilde{e}_m\}$	Encrypted image–text embedding set
$E_{svd} = \{e_1^s, e_2^s, \dots, e_m^s\}$	Incomplete image–text embedding matrix	$\tilde{E}_{svd} = \{\tilde{e}_1^s, \tilde{e}_2^s, \dots, \tilde{e}_m^s\}$	Encrypted incomplete image–text embedding matrix
E_Q	Embedding of query	\tilde{E}_Q	Query trapdoor

5.2.2. The Operation of the Encryption Model in the Level I Security Mode

Assuming the original dimension of the data feature embeddings is n , the dimension of the image–text embedding matrix E , which is formed by vertically concatenating E_t and E_I , is $m \times n$. The operation process of the encryption model in the level I security mode is as follows:

(1) $KeyGen(1^\lambda) \rightarrow S, M_1, M_2$: Key generation algorithm. DO inputs the security parameter λ and outputs a random index encryption key M . M is an invertible matrix with dimensions $(n + 1) \times (n + 1)$.

(2) $Enc_{data}(D) \rightarrow \tilde{D}, Enc_{data}(I) \rightarrow \tilde{I}$: Data encryption algorithm. DO uses text and image encryption algorithms to encrypt D and I into \tilde{D} and \tilde{I} . The specific encryption algorithms for text and images differ and will not be detailed here.

(3) $SVD(E) \rightarrow E_{svd}$: Embeddings matrix decomposition algorithm. PS uses the SVD algorithm to decompose the matrix E into three submatrices U, Σ , and V^T . The left singular

matrix $U(m \times m)$ is then multiplied by the singular-value matrix $\Sigma(m \times n)$. Additionally, to meet the retrieval functionality requirements, we need to extend $(U \times \Sigma)$ by one dimension to store the file number corresponding to each feature embedding, with the numbering range from 1 to m . Thus, the dimension of $(U \times \Sigma)$ should be $m \times (n + 1)$, and it is referred to as the incomplete feature embedding matrix $E_{svd} = \{e_1^s, e_2^s, \dots, e_m^s\}$.

(4) $Enc_{index}(E_{svd}, M) \rightarrow \widetilde{E}_{svd}$: Incomplete feature embeddings matrix encryption algorithm. PS uses the index encryption key M to encrypt the incomplete embedding matrix E_{svd} , resulting in the encrypted incomplete feature embedding matrix \widetilde{E}_{svd} with dimensions $m \times (n + 1)$. Specifically, $E_{svd} \times M = (U \times \Sigma) \times M = \widetilde{E}_{svd}$. \widetilde{E}_{svd} is then uploaded to the cloud server.

(5) $Enc_{index}(E_Q) \rightarrow \widetilde{E}_Q$: Trapdoor construction algorithm. Upon receiving the query from DU , PS extracts its feature $E_Q(1 \times n)$ and uses M^{-1} and V^T to encrypt it into the query trapdoor \widetilde{E}_Q . The original dimension of V^T should be $n \times n$. To ensure compatibility between the query and the incomplete feature embeddings matrix during computation, V^T needs to be extended by one dimension in the row direction, with all values in this dimension set to 0, resulting in a final dimension of $(n + 1) \times n$ for V^T . The specific computation process is: $M^{-1} \times V^T \times E_Q^T = \widetilde{E}_Q$.

In level I security mode, if DO wants to add new data, they need to use the SVD algorithm to decompose the new matrix composed of the feature embeddings of the original data and the feature embeddings of the new data. The more frequently DO adds new data, the more computational resources are consumed for each repeated decomposition. Inspired by the method in [?], we adopt the following approach to solve this problem.

For the original feature embedding matrix E , we have $SVD(E^T) = V\Sigma U^T$. Suppose the feature embedding matrix of the newly added data is E_1 ; then the new complete feature embedding matrix is $E' = [E, E_1]$. The process of performing SVD decomposition on E' is shown in Equation (4):

$$E' = [E^T, E_1^T] = [V\Sigma U^T, E_1^T] = V[\Sigma, V^T E_1^T] \begin{bmatrix} U^T & 0 \\ 0 & I \end{bmatrix} = VW \begin{bmatrix} U^T & 0 \\ 0 & I \end{bmatrix} = V(U_w \Sigma_w V_w^T) \begin{bmatrix} U^T & 0 \\ 0 & I \end{bmatrix} = (VU_w) \Sigma_w \left(\begin{bmatrix} U & 0 \\ 0 & I \end{bmatrix} V_w \right)^T \quad (4)$$

In Equation (4), $W = [\Sigma, V^T E_1^T]$ and $SVD(W) = U_w \Sigma_w V_w^T$. Finally, E' is decomposed into $E' = U_E \Sigma_E V_E^T$, where $U_E = VU_w$, $\Sigma_E = \Sigma_w$, and $V_E = \begin{bmatrix} U & 0 \\ 0 & I \end{bmatrix} V_w$.

In the decomposition process of E' described above, we do not need to perform SVD decomposition on the newly expanded feature embedding matrix E' . Instead, we only need to perform SVD decomposition on the newly added feature embedding matrix E_1 and then combine the results with the original matrix E to obtain the decomposition of E' . This approach significantly saves computational resources during the process of adding new data.

5.2.3. The Operation of the Encryption Model in the Level II Security Mode

The operation process of the encryption model in the level II security mode is as follows. In this mode, we use the asymmetric-scalar-product-preserving encryption technique from the MRSE [?], which is adapted for computing the similarity between vectors through inner product calculation, to encrypt the feature embeddings of the data.

(1) $KeyGen(1^\lambda) \rightarrow S, M_1, M_2$: Key generation algorithm. DO inputs the security parameter λ and outputs a randomly generated vector S with dimensions $1 \times (n + 2)$ and encryption matrices M_1 and M_2 with dimensions $(n + 3) \times (n + 3)$; each element in S is a random 0 or 1 value.

(2) $Enc_{data}(D) \rightarrow \widetilde{D}, Enc_{data}(I) \rightarrow \widetilde{I}$: Data encryption algorithm. DO uses text and image encryption algorithms to encrypt D and I into \widetilde{D} and \widetilde{I} , respectively.

(3) $Enc_{index}(e_i, M_1, M_2, \varepsilon_i) \rightarrow \tilde{E}$: Feature embedding encryption algorithm. PS generates a random number ε_i for the i -th feature embedding e_i in E and convert e_i into $\bar{e} = (e_i, \varepsilon_i, 1, i)$, with a dimension of $n + 3$. Next, using S , \bar{e} is split into e'_i and e''_i . The splitting principle is: for $j \in [1, n + 2]$, if $S[j] = 0$, then $e'_i[j] = e''_i[j]$; otherwise, $e'_i[j] + e''_i[j] = e_i[j]$ (the splitting ratio is also random). The final encrypted feature embedding for the i -th entry is: $\tilde{e}_i = (e'_i M_1, e''_i M_2)$. Then, PS uploads \tilde{e}_i to CS to build the index structure.

(4) $Enc_{index}(E_Q, r, t) \rightarrow \tilde{E}_Q$: Trapdoor construction algorithm. PS receives the query Q from DU and extracts its feature E_Q , generating random numbers r and t . Then, E_Q is converted into $\bar{E}_Q = (rE_Q, r, t, 0)$, with a dimension of $n + 3$. Using S , \bar{E}_Q is split into E'_Q and E''_Q . The splitting principle is: for $j \in [1, n + 2]$, if $S[j] = 1$, then $E'_Q[j] = E''_Q[j]$; otherwise, $E'_Q[j] + E''_Q[j] = E_Q[j]$ (the splitting ratio is also random). The final trapdoor is $\tilde{E}_Q = (M_1^{-1}(E'_Q)^T, M_2^{-1}(E''_Q)^T)$.

5.3. Retrieval Model

5.3.1. Index Building and Search Process

The retrieval model of PITR is deployed on CS . It uses each encrypted feature embedding \tilde{E} uploaded by the PS as a vector node to construct a hierarchical graph-based index structure. When a \tilde{E}_Q is received from PS , it is used as a query point in the index structure for the approximate nearest neighbor search. CS then returns the k most relevant results. The detailed processes of index construction and search are shown in Algorithm 1 and Algorithm 2, respectively.

The index construction algorithm is achieved by continuously inserting new element nodes into the established index structure. Compared to traditional ciphertext retrieval schemes, this algorithm is more accommodating to node update operations. To delete a node, simply make it invisible to other nodes. However, if the number of nodes that need updating reaches a threshold, the index structure must be rebuilt.

Algorithm 1 Index structure construction

Input: Encrypted feature embedding set: $\tilde{E} = \{\tilde{e}_1, \tilde{e}_2, \dots, \tilde{e}_m\}$.

Output: Hierarchical graph-based index structure $Index_{hnsw}$.

Preparatory:

$SearchLayer(s, en, ef, l)$: start from en ; find the ef nearest node to s at layer l .

$GetNeighbors(l, s, m, W)$: select the m nearest nodes to s from the set W at level l .

$AddConnections(M, s)$: add bidirectional connections between M and s .

$RandomLayer(s)$: assign layers to s using random function.

W : current nearest neighbor element set. m : the number of connections each node needs to establish with other nodes. ef : the size of the dynamic candidate set.

- 1: $l_1 \leftarrow RandomLayer(\tilde{e}_1)$
 - 2: Insert the first node \tilde{e}_1 from \tilde{E} into the l_1 layer.
 - 3: **for** \tilde{e} in $\tilde{E} - \tilde{e}_1$ **do**
 - 4: $l_e \leftarrow RandomLayer(\tilde{e})$
 - 5: **for** l in $\{l_{top}, \dots, l_e + 1\}$ **do**
 - 6: $W \leftarrow SearchLayer(\tilde{e}, en, ef = 1, l)$
 - 7: $en \leftarrow$ the nearest node to \tilde{e} in W
 - 8: **end for**
 - 9: **for** l in $\{l_{e+1}, \dots, 0\}$ **do**
 - 10: $W \leftarrow SearchLayer(s, en, ef, l)$
 - 11: $neighbors \leftarrow GetNeighbors(l, s, W, m)$
 - 12: $AddConnections(neighbors, \tilde{e})$
 - 13: **end for**
 - 14: **end for**
-

Algorithm 2 Search in index structure

Input: $Index_{hmsw}$ and query node q ; the number of the nearest neighbor nodes to be returned is k .

Output: The nearest k nodes to q .

- 1: **for** $l \leftarrow \{l_{top}, \dots, l_{bottom+1}\}$ **do**
- 2: $W \leftarrow SearchLayer(q, en, ef = 1, l)$
- 3: $en \leftarrow$ the nearest node to \tilde{e} in W
- 4: **end for**
- 5: \triangleright Starting from en obtained from the previous layer, search the bottom layer to obtain ef nearest neighbor nodes to q .
- 6: $W \leftarrow SearchLayer(q, en, ef, l = 0)$
- 7: return the nearest k nodes to q from W

5.3.2. Similarity Calculation

When CS performs the approximate nearest neighbor search using the query trapdoor \tilde{E}_Q in $Index_{hmsw}$, the distance calculation between vector nodes is based on cosine distance. Its reciprocal is the cosine similarity between the feature embeddings corresponding to the nodes. Since data features have been normalized during extraction, it is not necessary to divide by the vector modulus when calculating similarity. The following are the similarity calculation processes under the two security modes:

(1) **Level I security mode:** To ensure that the query trapdoor has a certain level of indistinguishability, a random number r is added during the trapdoor construction. This makes the trapdoors generated from the same query not identical, but it does not affect the consistency of the query results. The specific process is shown in Equation (5):

$$Enc_{index}(E_Q, r) \rightarrow \tilde{E}_Q = M^{-1} \times V^T \times E_Q^T \cdot r \quad (5)$$

The encrypted incomplete feature embedding matrix is $\tilde{E}_{svd} = \{\tilde{e}_1^s, \tilde{e}_2^s, \dots, \tilde{e}_m^s\}$, where \tilde{e}_i^s represents the i -th embedding in \tilde{E}_{svd} and is also the i -th vector node in $HNSW_{index}$. The computation process between the trapdoor and \tilde{e}_i^s is shown in Equation (6):

$$\begin{aligned} \tilde{e}_i^s \times \tilde{E}_Q &= e_i^s \times M \times M^{-1} \times V^T \times E_Q^T \cdot r \\ &= e_i^s \times V^T \times E_Q^T \cdot r = e_i \times E_Q^T \cdot r \end{aligned} \quad (6)$$

where e_i represents the i -th feature embedding in the original image–text feature embedding matrix E . The resulting scalar $e_i \times E_Q^T \cdot r$ represents the cosine similarity between the query vector E_Q and the node vector e_i .

(2) **Level II security mode:** The similarity calculation formula between the i -th embedding \tilde{e}_i in \tilde{E} and the query trapdoor \tilde{E}_Q is shown in Equation (7):

$$\begin{aligned} \tilde{e}_i \cdot \tilde{E}_Q &= \{e'_i M_1, e''_i M_2\} \cdot \{M_1^{-1}(E_Q)^T, M_2^{-1}(E_Q)^T\} \\ &= e'_i \cdot E_Q + e''_i \cdot E_Q \\ &= \tilde{e}_i \cdot E_Q = r(e_i \cdot E_Q + \varepsilon_i) + t \end{aligned} \quad (7)$$

5.4. Model Security Analysis

(1) **Level I security mode:** In the level I security mode, we perform SVD decomposition on the image–text feature embedding matrix E and upload a portion of the decomposed incomplete embeddings matrix to CS to build the index structure. The purpose of this is if the encryption key M is inadvertently obtained by CS, CS can only compute the original incomplete feature embedding index stored on it. However, since these incomplete feature embeddings are only a part of the complete feature embeddings matrix, CS cannot infer the

similarity relationships between different feature embeddings or the associations between feature embeddings and their corresponding plaintext data.

In level I mode, after decomposing the expanded feature embedding matrix E' formed by the addition of new user data, the decomposed partial results need to be encrypted and uploaded to CS, where the index structure is reconstructed. Therefore, level I mode is more suitable for storing data that remain fixed and unchanged over the long term.

(2) **Level II security mode:** By decomposing the feature embedding set of image and text data into two parts and adding some random numbers during the encryption process, we effectively ensure data privacy. Additionally, random numbers are also added to the user's query embedding, so even for the same query made twice, the resulting trapdoors are different. This can resist access pattern attacks from CS and prevent it from linking user queries with outsourced data. However, the final similarity calculation results will also be affected by these added random numbers, which can impact the accuracy of the retrieval tasks. Therefore, the value of the added random numbers should be appropriate to ensure effective data privacy protection while minimizing their impact on retrieval accuracy.

6. Experiment and Performance Analysis

In this section, we test the performance of the feature extraction model and the retrieval model in our PITR scheme, including capability testing for zero-shot image classification and cross-modal retrieval tasks on image–text data in both general and specialized domains. Our experimental environment is as follows: system version: Ubuntu 20.04, software environment: Python 3.10, CUDA 12.2; hardware environment: two NVIDIA GeForce RTX 4090D GPUs with a total of 48GB VRAM and a CPU: 30 vCPU Intel(R) Xeon(R) Platinum 8474C.

6.1. Zero-Shot Image Classification Task

6.1.1. Image Classification in the General Data Domain

The text feature extractor for PITR uses RoBERTa-wwm-Large with 24 layers and 24 heads in the multi-head attention module. The visual feature extractor uses ViT-H/14 with 32 layers and 16 heads in the multi-head attention module. The dimensions of the text and image embeddings are both 1024, and the total number of model parameters is 958M.

Based on this, we added 15 pseudo-prompts and a prompt encoder using an LSTM network to the input end of PITR. We then trained the model using the manually corrected Chinese versions of the Caltech-101, CIFAR-100, and CIFAR-10 general domain datasets. Table 2 shows the accuracy comparison of PITR with other multimodal pre-trained models in both general domain and specialized domain datasets for zero-shot image classification tasks. The comparison models selected were BriVL, Wukong, and CN-CLIP.

Table 2. Performance comparison for zero-shot image classification.

Scheme	Caltech-101	Cifar-100	Cifar-10	Pets	Food	Flower	DTD	EuroSAT
BriVL	-	35.9	72.3	-	-	-	-	-
Wukong	-	77.1	95.4	-	-	-	-	-
CN-CLIP	90.6	79.7	96.0	83.5	74.6	68.4	51.2	52.0
PITR	92.1	81.4	96.5	87.0	83.3	81.4	79.6	70.0

6.1.2. Image Classification in the Specialized Data Domain

To test the PITR's ability to learn specialized domain knowledge, we incorporated different specialized adapters according to the specific professional domains and then trained the model using the Chinese versions of specialized domain datasets, which belong to categories including pets, food, flowers, cars, surface textures, and satellite geographic images. The experimental results are shown in Table 2.

6.2. Image–Text Retrieval in General Domain Data

We trained on the general domain image–text dataset Flickr30K-CN using a combination of general adapter and contrastive learning methods. Additionally, for the text and image feature embeddings, we added the original data’s corresponding ID to the last dimension of each embedding before encrypting and uploading them to CS. Then we used the HNSW algorithm to build the index for retrieval. For the feature embeddings of user queries, we added a dimension to store a zero value, ensuring it does not affect the similarity calculations between the query and index vectors.

Table 3 shows the recall rates of PITR and other pre-trained large models for image-to-text retrieval. Assume the total number of queries is m , and for a given query, the number of relevant returned results is k . If there is at least one correct result among the k results, the score for that query is 1; otherwise, it is 0. The recall rate $Recall@k$ in the image retrieval domain is defined as in Equation (8).

$$Recall@k = \frac{1}{m} \sum_m^{i=1} score_i \quad (8)$$

Table 3. Comparison of the recall rates when performing the image-to-text task.

Scheme	R@1	R@5	R@10
Wukong	76.1	94.8	97.5
R2D2	77.6	96.7	98.9
CN-CLIP	81.6	97.5	98.8
PITR	82.8	97.3	99.2

We further tested the model’s performance in text-to-image retrieval tasks using the COCO-CN dataset for training and testing. The training method also utilized a combination of the general adapter and contrastive learning. Table 4 presents a comparison of recall rates for text-to-image retrieval tasks.

Table 4. Comparison of the recall rates when performing the text-to-image task.

Schemes	R@1	R@5	R@10
Wukong	53.4	80.2	90.1
R2D2	56.4	85.0	93.1
CN-CLIP	69.2	89.9	96.1
PITR	70.4	91.1	96.9

6.3. Image–Text Retrieval in Specialized Domain Data

We continued to use the model from the specialized domain image classification task and trained it using contrastive learning on specialized domain multimedia datasets, with category labels representing image captions. Tables 5 and 6 show the performance of PITR in image-to-text and text-to-image retrieval tasks, respectively.

Table 5. Image-to-text experiment on specialized domain datasets.

Dataset	R@1	R@5	R@10
Pets	87.0	96.3	99.7
DTD	79.6	86.2	95.8
EuroSAT	70.0	88.4	100
Flowers	81.7	89.9	96.2

Table 6. Text-to-image experiment on specialized domain datasets.

Dataset	R@1	R@5	R@10
Pets	88.1	98.5	100
DTD	83.0	91.9	98.4
EuroSAT	71.8	85.2	93.7
Flowers	84.4	96.3	99.8

6.4. Retrieval Speed Experiment

We tested the speed advantage of PITR compared to other similar schemes. The comparison schemes selected were $SSSW_2$ [?] and $SSRB_2$ [?], both of which generate feature embeddings with a dimension of 768, and PITR generates expanded feature vectors with a dimension of 1027. Figure 8 shows the change in time consumption for PITR during the feature extraction and indexing construction process as the number of files increases (each image and text sentence represents a file). As shown in the figure, PITR's time consumption in this process is on the order of $1e1$ (1×10^1) seconds, while the other two schemes' time consumption for the same process is on the order of $1e3$ (1×10^3) seconds [?]. Figure 9 shows the indexing construction process as the number of files increases.

Figure 10 shows the change in time consumption of PITR for performing nearest neighbor searches for query trapdoors on an established index structure as the number of files increases. Some important parameters for the HNSW algorithm are $ef_{construction} = 200$, $M = 16$, and $k = 5$. As shown in the figure, compared to the other two schemes, PITR's time consumption during the search process remains very stable as the number of files increases. Additionally, when the number of files is large, the speed advantage of PITR becomes more obvious.

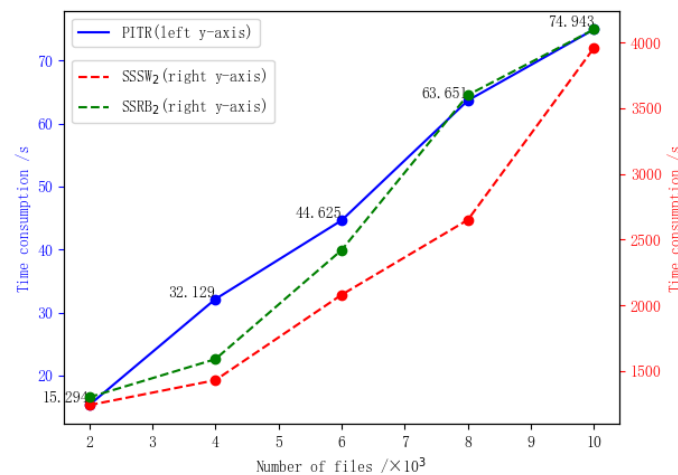


Figure 8. The time consumption for feature extraction and index construction varies with the number of files.

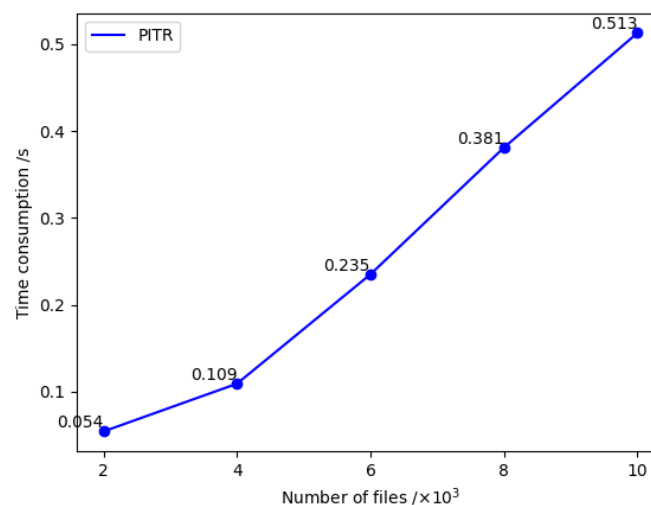


Figure 9. The time consumption for index construction varies with the number of files.

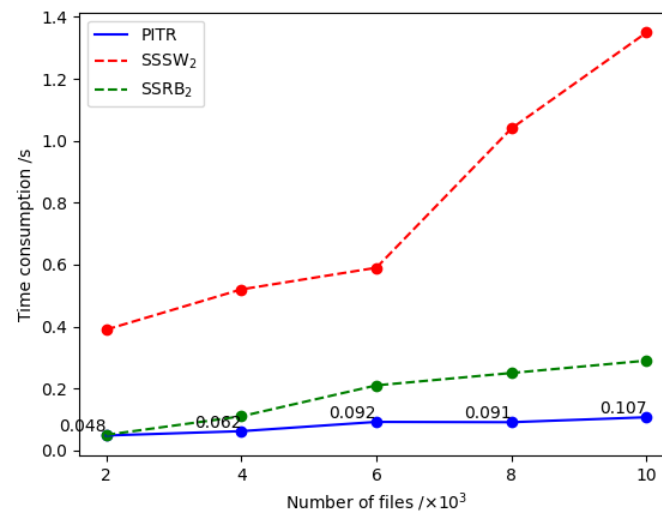


Figure 10. Comparison of the time consumption during the search process as the number of files increases.

7. Conclusions

This paper proposes an efficient privacy-preserving cross-modal image–text retrieval scheme: Pitr. By introducing the general adapter and the specialized adapter into the frozen multimodal pre-trained model, we enhanced the model’s ability to extract global features from multimodal data in both general and specific domains with minimal training resource overhead, thereby meeting users’ diverse cross-modal retrieval needs. Experiments show that, compared to previous methods, Pitr improves the average accuracy of zero-shot image classification tasks by approximately 1.23% and 14.32% in general and specialized domain datasets, respectively. Additionally, the average recall rates for image-to-text and text-to-image retrieval tasks in general domain datasets are improved by approximately 0.47% and 1.07%, respectively. In terms of privacy security, Pitr employs a two-level security model, utilizing ASPE technology suitable for inner product calculation and distributed index storage technology to meet users’ varying privacy protection requirements. For retrieval, Pitr uses the HNSW algorithm to construct encrypted indexes and perform searches, improving the efficiency of retrieval among massive high-dimensional vectors.

Future work includes: (1) investigating how to achieve more efficient and convenient data updating capabilities under the level I security mode, (2) exploring the application of homomorphic encryption techniques in key parts to further enhance security and examining alternatives to Intel SGX to reduce the risk of data leakage due to hardware attacks, (3) researching more robust end-to-end encryption schemes and secure key management protocols to strengthen communication security among various entities, and (4) exploring how to construct a unified dataset containing more media types to realize a more powerful cross-modal retrieval system.

Author Contributions: Conceptualization, S.X. and K.Z.; methodology, S.X.; software, S.X.; validation, S.X., Y.S. and Y.X.; formal analysis, P.L.; investigation, X.Y.; resources, W.W.; data curation, B.Z.; writing—original draft preparation, S.X.; writing—review and editing, K.Z.; visualization, S.X.; supervision, Y.S., W.W. and P.L.; project administration, X.Y. and B.Z.; funding acquisition, K.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Fundamental Research Funds for the Central Universities [grant number 3282023012] and the Network Security Team Construction 2024 [grant number 3282024056].

Data Availability Statement: The data presented in this study are available on request from the author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Ferreira, B.; Rodrigues, J.; Leitao, J.; Domingos, H. Practical Privacy-Preserving Content-Based Retrieval in Cloud Image Repositories. *Cloud Comput. IEEE Trans.* **2019**, *7*, 784–798. [\[CrossRef\]](#)
2. Liu, D.; Shen, J.; Xia, Z.; Sun, X. A Content-Based Image Retrieval Scheme Using an Encrypted Difference Histogram in Cloud Computing. *Information* **2017**, *8*, 96. [\[CrossRef\]](#)
3. Xia, Z.; Jiang, L.; Ma, X.; Yang, W.; Ji, P.; Xiong, N. A Privacy-Preserving Outsourcing Scheme for Image Local Binary Pattern in Secure Industrial Internet of Things. *IEEE Trans. Ind. Inform.* **2019**, *16*, 629–638. [\[CrossRef\]](#)
4. Ma, W.; Zhou, T.; Qin, J.; Xiang, X.; Tan, Y.; Cai, Z. A privacy-preserving content-based image retrieval method based on deep learning in cloud computing. *Expert Syst. Appl.* **2022**, *203*, 117508. [\[CrossRef\]](#)
5. Feng, Q.; Li, P.; Lu, Z.; Li, C.; Wang, Z.; Liu, Z.; Duan, C.; Huang, F.; Weng, J.; Yu, P.S. Evit: Privacy-preserving image retrieval via encrypted vision transformer in cloud computing. *arXiv* **2024**, arXiv:2208.14657. [\[CrossRef\]](#)
6. Zhang, L.; Jung, T.; Feng, P.; Liu, K.; Liu, Y. PIC: Enable Large-Scale Privacy Preserving Content-Based Image Search on Cloud. In Proceedings of the International Conference on Parallel Processing, Beijing, China, 1–4 September 2015.
7. Bellafqira, R.; Coatrieux, G.; Bousslimi, D.; Quelled, G. Content-based image retrieval in homomorphic encryption domain. In Proceedings of the 2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Milan, Italy, 25–29 August 2015; pp. 2944–2947.
8. Gilad-Bachrach, R.; Dowlin, N.; Laine, K.; Lauter, K.; Naehrig, M.; Wernsing, J. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In Proceedings of the International Conference on Machine Learning, New York City, NY, USA, 19–24 June 2016; pp. 201–210.
9. Juvekar, C.; Vaikuntanathan, V.; Chandrakasan, A. Gazelle: A Low Latency Framework for Secure Neural Network Inference. *arXiv* **2018**, arXiv:1801.05507.
10. Cheng, B.; Zhuo, L.; Bai, Y.; Peng, Y.; Zhang, J. Secure index construction for privacy-preserving large-scale image retrieval. In Proceedings of the 2014 IEEE Fourth International Conference on Big Data and Cloud Computing, Sydney, NSW, Australia, 3–5 December 2014; pp. 116–120.
11. Li, Y.; Ma, J.; Miao, Y.; Wang, Y.; Liu, X.; Choo, K.K.R. Similarity search for encrypted images in secure cloud computing. *IEEE Trans. Cloud Comput.* **2020**, *10*, 1142–1155. [\[CrossRef\]](#)
12. Huang, J.; Luo, Y.; Xu, M.; Fu, S.; Huang, K. Accelerating privacy-preserving image retrieval with multi-index hashing. In Proceedings of the 2022 IEEE/ACM 7th Symposium on Edge Computing (SEC), Seattle, WA, USA, 5–8 December 2022; pp. 492–497.
13. Li, J.; Li, D.; Savarese, S.; Hoi, S. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In Proceedings of the International Conference on Machine Learning, Honolulu, HI, USA, 23–29 July 2023; pp. 19730–19742.
14. Li, J.; Huang, Y.; Wei, Y.; Lv, S.; Liu, Z.; Dong, C.; Lou, W. Searchable symmetric encryption with forward search privacy. *IEEE Trans. Dependable Secur. Comput.* **2019**, *18*, 460–474. [\[CrossRef\]](#)
15. Wang, B.; Song, W.; Lou, W.; Hou, Y.T. Inverted index based multi-keyword public-key searchable encryption with strong privacy guarantee. In Proceedings of the 2015 IEEE Conference on Computer Communications (INFOCOM), Kowloon, Hong Kong, 26 April–1 May 2015; pp. 2092–2100.
16. Wang, B.; Hou, Y.; Li, M.; Wang, H.; Li, H. Maple: Scalable multi-dimensional range search over encrypted cloud data with tree-based index. In Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security, Kyoto, Japan, 4–6 June 2014; pp. 111–122.
17. Andola, N.; Prakash, S.; Yadav, V.K.; Raghav, Venkatesan, S.; Verma, S. A secure searchable encryption scheme for cloud using hash-based indexing. *J. Comput. Syst. Sci.* **2022**, *126*, 119–137. [\[CrossRef\]](#)
18. Liang, H.; Zhang, X.; Cheng, H. Huffman-code based retrieval for encrypted JPEG images. *J. Vis. Commun. Image Represent.* **2019**, *61*, 149–156. [\[CrossRef\]](#)
19. Xia, Z.; Jiang, L.; Liu, D.; Lu, L.; Jeon, B. BOEW: A content-based image retrieval scheme using bag-of-encrypted-words in cloud computing. *IEEE Trans. Serv. Comput.* **2019**, *15*, 202–214. [\[CrossRef\]](#)
20. Marcolla, C.; Sucasas, V.; Manzano, M.; Bassoli, R.; Fitzek, F.H.; Aaraj, N. Survey on fully homomorphic encryption, theory, and applications. *Proc. IEEE* **2022**, *110*, 1572–1609. [\[CrossRef\]](#)
21. Hu, S.; Zhang, L.Y.; Wang, Q.; Qin, Z.; Wang, C. Towards private and scalable cross-media retrieval. *IEEE Trans. Dependable Secur. Comput.* **2019**, *18*, 1354–1368. [\[CrossRef\]](#)
22. Zhu, L.; Song, J.; Yang, Z.; Huang, W.; Zhang, C.; Yu, W. DAP 2 CMH: Deep adversarial privacy-preserving cross-modal hashing. *Neural Process. Lett.* **2022**, *54*, 2549–2569. [\[CrossRef\]](#)
23. Wang, Z.; Qin, J.; Xiang, X.; Tan, Y.; Peng, J. A privacy-preserving cross-media retrieval on encrypted data in cloud computing. *J. Inf. Secur. Appl.* **2023**, *73*, 103440. [\[CrossRef\]](#)
24. Radford, A.; Kim, J.W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; et al. Learning transferable visual models from natural language supervision. In Proceedings of the International Conference on Machine Learning, Online, 18–24 July 2021; pp. 8748–8763.
25. Malkov, Y.A.; Yashunin, D.A. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *42*, 824–836. [\[CrossRef\]](#)

26. Liu, Y.; Fu, Z. Secure search service based on word2vec in the public cloud. *Int. J. Comput. Sci. Eng.* **2019**, *18*, 305–313. [[CrossRef](#)]
27. Fu, Z.; Wang, Y.; Sun, X.; Zhang, X. Semantic and secure search over encrypted outsourcing cloud based on BERT. *Front. Comput. Sci.* **2022**, *16*, 162802. [[CrossRef](#)]
28. Stewart, G.W. On the early history of the singular value decomposition. *SIAM Rev.* **1993**, *35*, 551–566. [[CrossRef](#)]
29. Shishkin, S.L.; Shalaginov, A.; Bopardikar, S.D. Fast approximate truncated SVD. *Numer. Linear Algebra Appl.* **2019**, *26*, e2246. [[CrossRef](#)]
30. Wong, W.K.; Cheung, D.W.I.; Kao, B.; Mamoulis, N. Secure kNN computation on encrypted databases. In Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data, Providence, RI, USA, 29 June–2 July 2009; pp. 139–152.
31. Lei, X.; Tu, G.H.; Liu, A.X.; Xie, T. Fast and secure knn query processing in cloud computing. In Proceedings of the 2020 IEEE Conference on Communications and Network Security (CNS), Avignon, France, 29 June–1 July 2020; pp. 1–9.
32. Cao, N.; Wang, C.; Li, M.; Ren, K.; Lou, W. Privacy-preserving multi-keyword ranked search over encrypted cloud data. *IEEE Trans. Parallel Distrib. Syst.* **2013**, *25*, 222–233. [[CrossRef](#)]
33. Yang, A.; Pan, J.; Lin, J.; Men, R.; Zhang, Y.; Zhou, J.; Zhou, C. Chinese clip: Contrastive vision-language pretraining in chinese. *arXiv* **2022**, arXiv:2211.01335.
34. Liu, X.; Zheng, Y.; Du, Z.; Ding, M.; Qian, Y.; Yang, Z.; Tang, J. GPT understands, too. *arXiv* **2023**, arXiv:2103.10385. [[CrossRef](#)]
35. Lester, B.; Al-Rfou, R.; Constant, N. The power of scale for parameter-efficient prompt tuning. *arXiv* **2021**, arXiv:2104.08691.
36. Yu, Y.; Si, X.; Hu, C.; Zhang, J. A review of recurrent neural networks: LSTM cells and network architectures. *Neural Comput.* **2019**, *31*, 1235–1270. [[CrossRef](#)]
37. Desai, M.; Shah, M. An anatomization on breast cancer detection and diagnosis employing multi-layer perceptron neural network (MLP) and Convolutional neural network (CNN). *Clin. eHealth* **2021**, *4*, 1–11. [[CrossRef](#)]
38. Liu, H.; Tam, D.; Muqeeth, M.; Mohta, J.; Huang, T.; Bansal, M.; Raffel, C.A. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 1950–1965.
39. Abdi, H.; Williams, L.J. Principal component analysis. *Wiley Interdiscip. Rev. Comput. Stat.* **2010**, *2*, 433–459. [[CrossRef](#)]
40. Anowar, F.; Sadaoui, S.; Selim, B. Conceptual and empirical comparison of dimensionality reduction algorithms (pca, kpca, lda, mds, svd, lle, isomap, le, ica, t-sne). *Comput. Sci. Rev.* **2021**, *40*, 100378. [[CrossRef](#)]
41. Zhou, X.; He, J.; Huang, G.; Zhang, Y. SVD-based incremental approaches for recommender systems. *J. Comput. Syst. Sci.* **2015**, *81*, 717–733. [[CrossRef](#)]
42. Zhang, K.; Xu, S.; Li, P.; Zhang, D.; Wang, W.; Zou, B. CRE: An Efficient Ciphertext Retrieval Scheme Based on Encoder. In Proceedings of the International Conference on Neural Information Processing; Springer: Berlin/Heidelberg, Germany, 2023; pp. 117–130.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.