Binary Embedding-based Retrieval at Tencent

Yukang Gan*
brucegan@tencent.com
ARC Lab, Tencent PCG

Yixiao Ge*
yixiaoge@tencent.com
ARC Lab, Tencent PCG

Chang Zhou*
chanzhou@tencent.com
Tencent Video, PCG

Shupeng Su
pennsu@tencent.com
ARC Lab, Tencent PCG

Zhouchuan Xu
atuerxu@tencent.com
Tencent Search, PCG

Xuyuan Xu
evanxyxu@tencent.com
Tencent Video, PCG

Quanchao Hui
andrewshui@tencent.com
Tencent Search, PCG

Xiang Chen
joshuaxchen@tencent.com
Tencent Search, PCG

Yexin Wang
yexinwang@tencent.com
Tencent Video, PCG

Ying Shan
yingsshan@tencent.com
ARC Lab, Tencent PCG
Tencent Search, PCG

ABSTRACT

Large-scale embedding-based retrieval (EBR) is the cornerstone of search-related industrial applications. Given a user query, the system of EBR aims to identify relevant information from a large corpus of documents that may be tens or hundreds of billions in size. The storage and computation turn out to be expensive and inefficient with massive documents and high concurrent queries, making it difficult to further scale up.

To tackle the challenge, we propose a binary embedding-based retrieval (BEBR) engine equipped with a recurrent binarization algorithm that enables customized bits per dimension. Specifically, we compress the full-precision query and document embeddings, formulated as float vectors in general, into a composition of multiple binary vectors using a lightweight transformation model with residual multi-layer perception (MLP) blocks. The bits of transformed binary vectors are jointly determined by the output dimension of MLP blocks (termed m) and the number of residual blocks (termed u), *i.e.*, $m \times (u+1)$. We can therefore tailor the number of bits for different applications to trade off accuracy loss and cost savings. Importantly, we enable task-agnostic efficient training of the binarization model using a new embedding-to-embedding strategy, *e.g.*, only 2 V100 GPU hours are required by millions of vectors for training. We also exploit the compatible training of binary embeddings so that the BEBR engine can support indexing among multiple embedding versions within a unified system. To further realize efficient search, we propose Symmetric Distance Calculation (SDC) to achieve lower response time than Hamming codes. The technique exploits Single Instruction Multiple Data (SIMD) units widely available in current CPUs.

We successfully employed the introduced BEBR to web search and copyright detection of Tencent products, including Sogou, Tencent Video, QQ World, *etc.* The binarization algorithm can be seamlessly generalized to various tasks with multiple modalities, for instance, natural language processing (NLP) and computer vision (CV). Extensive experiments on offline benchmarks and online A/B tests demonstrate the efficiency and effectiveness of our method, significantly saving 30% $\sim$ 50% index costs with almost no loss of accuracy at the system level[1].

CCS CONCEPTS

• **Information systems** → **Retrieval models and ranking**; **Search index compression**; **Retrieval efficiency**; • **Computing methodologies** → **Learning latent representations**; *Learning paradigms*; *Visual content-based indexing and retrieval.*

KEYWORDS

embedding-based retrieval, embedding binarization, backward compatibility

ACM Reference Format:
Yukang Gan, Yixiao Ge, Chang Zhou, Shupeng Su, Zhouchuan Xu, Xuyuan Xu, Quanchao Hui, Xiang Chen, Yexin Wang, and Ying Shan. 2023. Binary Embedding-based Retrieval at Tencent. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3580305.3599782

1 INTRODUCTION

With the development of deep learning, embedding-based retrieval (EBR) achieves great advances in real-world applications, such as web search [41], social search [22], e-commerce search [25], *etc.* Generally speaking, a typical industrial search-related system is

*The authors contributed equally to this work.

KDD '23, August 6–10, 2023, Long Beach, CA, USA

ACM ISBN 979-8-4007-0103-0/23/08…$15.00
https://doi.org/10.1145/3580305.3599782

[1]Code is publicly available at https://github.com/TencentARC/BEBR.

composed of a "recall-rerank" architecture (as demonstrated in Figure 1), in which the efficiency of the recall module with EBR algorithms is the bottleneck of the whole system as it needs to process massive documents. Unlike conventional inverted index-based term matching [35] that measures similarity through lexical analysis, EBR represents queries and documents as dense feature vectors. Given a query, EBR retrieves a set of relevant documents according to their embedding similarities in the latent space. The enormous scale of documents and high concurrent queries pose great challenges to an industrial EBR system, including retrieval latency, computation cost, storage consumption, and embedding upgrades.

There are previous attempts to develop more efficient EBR systems with advanced ANN (Approximate Nearest Neighbor) algorithms, *e.g.*, HNSW [28]. Though the achievements in saving computations, they need elaborate designs to be adapted and plugged into existing systems. Given the large number and variety of EBR systems for Tencent products, the development costs of upgrading all the existing ANN algorithms are non-negligible and even unaffordable. Toward this end, we focus on the most fundamental component of EBR, that is, embedding, also known as representation learning in the deep learning community. Properly compressed embeddings are compatible with mainstream ANN algorithms and can be seamlessly integrated into existing EBR systems.

In this work, we propose a binary embedding-based retrieval (BEBR) engine that has several appealing benefits: (i) customizable embedding compression rates to receive a trade-off between accuracy and costs; (ii) a task-agnostic and modal-agnostic efficient training paradigm for easy generalization and data security protection; (iii) a free embedding upgrading mechanism with backward compatibility, *i.e.*, no need to refresh the index. BEBR has been well deployed on multiple Tencent products equipped with various ANN algorithms (*e.g.*, IVF [30], HNSW [28]) with almost no accuracy loss and 30∼50% cost savings at the system level.

Specifically, inspired by recurrent binary embeddings [36] that progressively refine a base binary vector with binary residual vectors to meet task accuracy, BEBR develops a universal binarization algorithm with state-of-the-art performances across modalities. Rather than the simple linear transformations used in [36], BEBR adopts multilayer perception (MLP) blocks with non-linear layers (*i.e.*, ReLU [1]) for both binarization (float→binary) and reconstruction (binary→float) in the recurrent learning paradigm. As illustrated in Figure 3, the binarization, reconstruction, and residual blocks together form the recurrent binarization module with a customized number of loops, *i.e.*, bits per dimension. The recurrent binary embeddings with richer representations are much more discriminative than ordinary hash vectors [9].

In previous works, the binarization (or hashing) module is usually optimized end-to-end with the backbone network, *e.g.*, CNNs [38] for vision, and Transformers [33] for text. The training is expensive considering the heavy backbone models for accurate retrieval. We, therefore, introduce an efficient training paradigm that requires only floating-point vectors as input. The lightweight binarization module is trained individually without accessing the backbone models, forming a universal training procedure for all the modalities and tasks. To enable effective representation learning in such an embedding-to-embedding paradigm, we use contrastive learning with queue-based hard negative mining as the training objectives.

Besides the index costs, large-scale EBR systems heavily suffer from the computational overhead required by embedding model upgrades. In particular, all the embeddings in the index need to be re-extracted before the deployment of a new model, which may take weeks or even months for industrial applications. Thanks to the pioneering work in compatible representation learning [21, 37], we take the first step to investigate compatible training of binary embeddings. Equipped with backward-compatible learning, our BEBR engine is able to harvest the benefit of the new model immediately, *i.e.*, the queries encoded by the new model can be directly indexed among the old index.

We further propose Symmetric Distance Calculation (SDC) of recurrent binary embeddings, a novel technique that achieves significant speedup over the conventional Hamming-based distance calculation in [36]. SDC leverages the in-register cache to perform fast SIMD (Single Instruction Multiple Data) look-up instructions and is especially in favor of CPU-based computation platforms. Comprehensive experiments on public benchmarks, internal datasets, and online A/B tests on Tencent products fully demonstrate the effectiveness of our BEBR engine. It has been successfully deployed on almost all ranges of index-based applications in Tencent PCG, including web search (Sogou), video search, copyright detection, video recommendation, *etc.*

The contributions are four-fold.

- We propose a binary embedding-based retrieval (BEBR) engine that efficiently indexes among tens of billions of documents in Tencent products. The proposed method can be equipped with various ANN algorithms and integrated into existing systems seamlessly.
- BEBR drastically reduces both the memory and disk consumption while achieving superior retrieval performance with the benefit of tailored recurrent binarization and symmetric distance calculation.
- BEBR develops a universal training paradigm for all modalities without accessing the raw data and backbone networks, *i.e.*, the binary embeddings are trained efficiently in a task-agnostic embedding-to-embedding manner.
- BEBR enables backward-compatible upgrades of embedding models, that is, the new model can be immediately deployed without refreshing the index embeddings. We for the first time study compatible learning on binary embeddings.

2 RELATED WORK

2.1 Embedding-based Retrieval in Search

Representation learning for embedding-based retrieval has garnered significant attention in both academia and industry due to its remarkable success across various domains. For example, as a social search engine, Facebook learns semantic embeddings for personalized search, which serves an EBR system with ANN parameter tuning [22]. For e-commerce search, Taobao proposes a Multi-Grained Deep Semantic Product Retrieval (MGDSPR) [25] system to capture the relation between user query semantics and his/her personalized behaviors. Amazon develops a Siamese network to address the semantic gap problem for semantic product

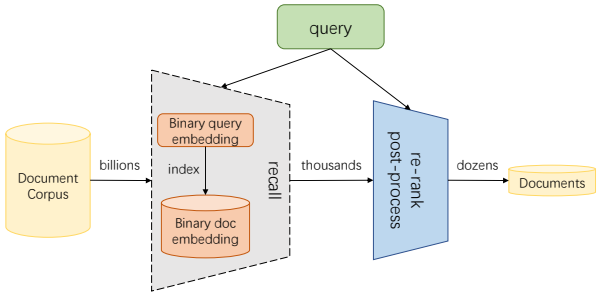

Figure 1: A brief structure of the Tencent search system, composed of the introduced binary embedding-based retrieval (BEBR) engine for recall and a re-rank post-process layer.

retrieval [31]. For web search, Google adopts zero-shot heterogeneous transfer learning in the recommendation system to improve search performance [41]. While none of the aforementioned methods studies the trade-off between performance and costs in EBR implementation, this paper discusses the binary embedding-based retrieval system, which can achieve near-lossless performance with significant cost reduction.

2.2 ANN Methods

Considerable research efforts have been dedicated to the development of efficient Approximate Nearest Neighbor (ANN) algorithms. Among these approaches, certain methods construct graphs from datasets to mitigate the need for exhaustive searches. In this context, each vertex in the graph corresponds to a specific data point. Alternatively, some algorithms encode the embeddings into compact codes to minimize memory consumption and expedite distance calculations.

Graph-based methods, in particular, exploit the utilization of the k-Nearest Neighbor graph to facilitate efficient navigation within the index. For instance, a proximity graph algorithm known as Navigable Small World (NSW) was proposed in [27], which incorporates navigable graphs. Moreover, Hierarchical NSW [28] introduces a controllable hierarchy, significantly improving scalability with a logarithmic complexity scaling. More recently, Navigating Spreading-out Graph (NSG) [16] introduces a novel graph structure that guarantees minimal search complexity, surpassing previous state-of-the-art approaches. Despite the impressive search performance achieved by these graph-based algorithms, they require more memory space and data pre-processing time compared to product quantization and hashing-based methods. Consequently, in scenarios involving frequent updates, constructing the index using graph-based algorithms on large datasets becomes impractical.

Product quantization (PQ) [23] is a technique that decomposes the space into a Cartesian product of low-dimensional subspaces and quantizes each subspace independently. Building upon this idea, Cartesian K-means (CKM) [32] and Optimized Product Quantizers (OPQ) [17] further enhance the sub-space decomposition by arbitrarily rotating and permutating vector components. Additionally, various quantization models [5, 6, 43] inspired by PQ have been introduced, offering lower quantization errors compared to PQ or OPQ. Notably, PQ Fast Scan [4] pioneers the utilization of Single Instruction, Multiple Data (SIMD) for Asymmetrical Distance Calculation (ADC) evaluation. Subsequent works [2, 3, 7] have proposed further optimizations of the quantization scheme, aiming to reduce search latency in indexed databases. Drawing inspiration from these ADC techniques, we propose a symmetrical distance calculation (SDC) method to facilitate efficient search in embedding-based retrieval (BEBR).

Learning to hash algorithms have gained significant popularity in recent times due to their computational and storage advantages. With the advancement of deep learning, numerous methods [9, 15, 26, 38] leverage the powerful capabilities of deep neural networks (DNNs) to learn complex hash functions and generate binary codes in an end-to-end manner. Rather than converting data into regular binary vectors, [36] proposes a recurrent binary embedding technique that achieves a balance between retrieval performance, speed, and memory requirements. This approach progressively adds a residual binary vector to a base binary vector. Additionally, a GPU-based k-nearest neighbor (K-NN) selection algorithm is implemented, enabling real-time exhaustive search on datasets containing billions of entries. In this paper, our focus is on achieving efficient and cost-effective embedding quantization. We utilize off-the-shelf floating-point-based embeddings as input to learn recurrent binary embeddings. Furthermore, we present an efficient method to calculate distances between recurrent binary embeddings using CPUs, which are the most commonly used computing devices in industrial retrieval systems.

2.3 Compatibility of Deep Neural Networks

Compatible representation learning has garnered significant attention in both industry and academia due to its ability to enable the comparison of embeddings across different models. This approach is particularly valuable as it helps reduce computation costs in embedding upgrades. Two main types of compatibility are commonly addressed: cross-model compatibility and backward compatibility. Cross-model compatibility learning focuses on training transformation modules that map embeddings from various models into a shared or common space. In the field of face recognition, R^3AN [10] was the first to address the issue of cross-model compatibility. They tackled this challenge by learning a transformation that converts source features into target features through a process involving reconstruction, representation, and regression. Building upon this, [39] introduced a unified representation learning framework that significantly enhances cross-model compatibility performance. In this framework, they devised a lightweight Residual Bottleneck Transformation (RBT) module and optimized it using a combination of classification loss, similarity loss, and KL-divergence loss. These efforts collectively contribute to improving the performance and effectiveness of cross-model compatibility in representation learning.

While cross-model compatibility focuses on handling embeddings from different models, backward compatibility shifts its attention to model updates, specifically when new models are trained with additional compatibility constraints. In this context, backward compatibility ensures that the new embeddings can be compared directly with the old embeddings without the need for any extra transformation processes. The pioneering work in this area is [37],

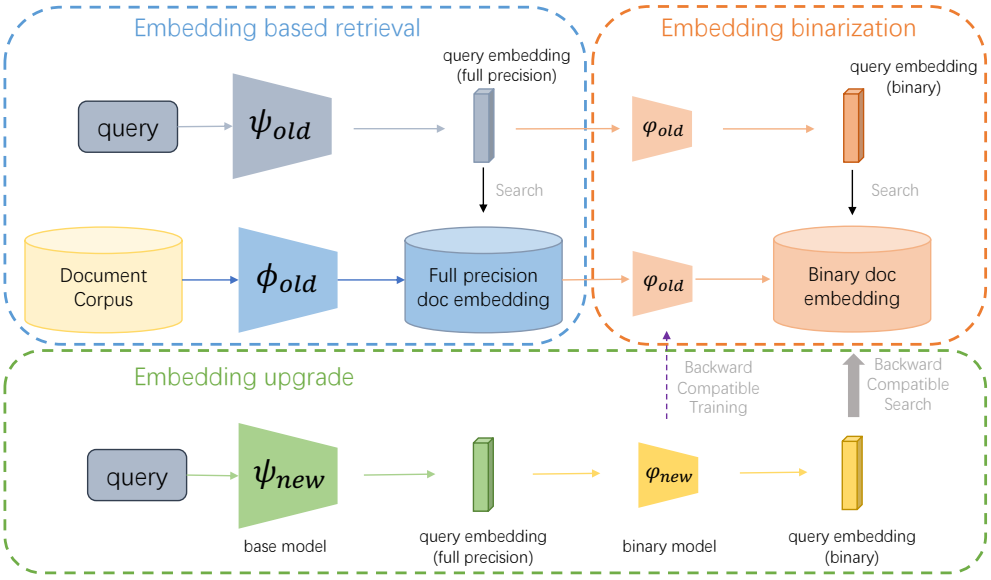

Figure 2: Our binary embedding-based retrieval (BEBR) framework. The full-precision float embeddings, extracted by the backbone networks, are transformed to recurrent binary vectors using a parametric binarization module φ in a task-agnostic embedding-to-embedding manner. BEBR enables backfill-free upgrades for the binarization model, that is, the new model can be immediately deployed for encoding better query embeddings without refreshing the index.

which utilizes backward compatibility for conducting model upgrades. It introduces an influence loss during the training of the new model to enable a direct comparison between the new and old embeddings. Subsequent works within the backward compatibility framework aim to enhance its performance through techniques such as hot-refresh backward-compatible model upgrades [42], asymmetric retrieval constraints [8], embedding cluster alignment loss [29], and neural architecture search [14]. In this paper, we employ backward compatibility training to learn backward compatible binary embeddings. To the best of our knowledge, this is the first application of compatible learning to binary embeddings.

3 BINARY EMBEDDING-BASED RETRIEVAL

3.1 Preliminary

Given a query q (generally a text in a web search or a video in copyright detection), an embedding-based retrieval (EBR) system aims to rank the documents $\{d_0, d_1, \cdots, d_n\}$ according to their similarities. There are two key factors in EBR, the embedding model(s) and the distance calculation metric $\mathcal{D}(\cdot,\cdot)$. The cosine similarity is widely used as $\mathcal{D}(\cdot,\cdot)$ for full precision embeddings (float vectors). Formally, the similarity between a certain query and a document is

$$\mathcal{S}_{\text{EBR}}(q, d_k) = \mathcal{D}\left(\psi(q), \phi(d_k)\right), \ \forall k \in \{1, \cdots, n\}, \tag{1}$$

where ψ and ϕ are embedding models for queries and documents. ψ and ϕ can be designed to be identical or distinct to handle different retrieval tasks with homogeneous or heterogeneous input [18]. For example, we may have ResNet [20] for image data and Transformer [13] for text. Without loss of generality, we consider homogeneous architecture (*i.e.*, ψ and ϕ are identical and both denoted as ϕ) in the following cases. To tackle the billion-level indexing at a moderate cost, we introduce binary embedding-based retrieval (BEBR) engine with much more efficient similarity calculation between queries and documents, such as

$$\mathcal{S}_{\text{BEBR}}(q, d_k) = \mathcal{D}\left(\phi \circ \varphi(q), \phi \circ \varphi(d_k)\right), \ \forall k \in \{1, \cdots, n\}, \tag{2}$$

where $\varphi(\cdot)$ is the binarization process and is generally realized by a parametric network. In the following sections, we will introduce the detailed designs of $\varphi(\cdot)$ in Section 3.2 and $\mathcal{D}(\cdot,\cdot)$ in Section 3.3.

3.2 Recurrent Binarization

3.2.1 Architecture. To tackle the problem of learning to binarization, the straightforward solution is to adopt hashing networks [44] ended up with a binarization function ρ, which plays an important role in converting float vectors into binary ones composed of either -1 or $+1$. In the forward pass, ρ is formulated as

$$\rho(x) \equiv \text{sign}(x) = \begin{cases} -1, & x \le 0 \\ 1, & x > 0 \end{cases}$$

Since the gradient of the sign function vanishes and thus cannot be back-propagated directly, [12] introduced a straight-through estimator (STE) that takes the gradient of the identity function instead, that is, $\rho'(x) = 1$ when $|x| \le 1$ and $\rho'(x) = 0$ otherwise. Conventional hashing methods generally convert the float vectors to binary embeddings once with some learnable layers and a binarization function introduced above. However, such methods suffer from unsatisfactory performance due to the limited representation ability of the hash codes that only have -1 or $+1$ values, *i.e.*, 1 bit per dimension. Thanks to the pioneering work [36], binary embeddings are able to be progressively refined with customized bits per dimension.

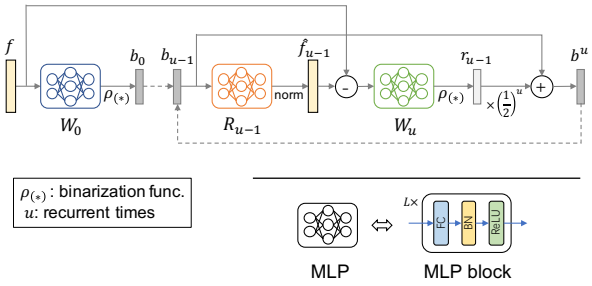

Figure 3: The architecture of recurrent binary embedding model φ. $\ominus$ and $\oplus$ denote minus and plus operations between two input embeddings, respectively.

Specifically, following the insight of using residual operations to gradually narrow the gap between original float vectors and the learned binary ones, we introduce a recurrent binarization module with customized loops, as demonstrated in Figure 3. There are three main components, including binarization block, reconstruction block, and residual block. The binarization block performs almost the same as conventional hashing networks, where a binary embedding b_0 is encoded from the float vector f as input: $b_0 = \rho(\boldsymbol{W}_0(f)) \in \{-1, 1\}^m$, where ρ is the binarization function, and $\boldsymbol{W}_0$ is the multi-layer perception (MLP) that consists of linear, batch normalization, and ReLU layers. The encoded binary embedding b_0 is then reconstructed back to float vectors, such as $\hat{f}_0 = \|\boldsymbol{R}_0(b_0)\|$, where $\boldsymbol{R}_0$ is also multi-layer perception (MLP). The residual between the original f and reconstructed $\hat{f}_0$, therefore, reflects the representation loss of the binarization process, which can be further narrowed by repeating the above steps to binarize the residual parts. The residual binary vector can be formulated as $r_0 = \rho(\boldsymbol{W}_1(f - \hat{f}_0))$, which is further added to the base binary vector b_0 via $b_1 = b_0 + \frac{1}{2}r_0$. The weight $\frac{1}{2}$ is chosen to ease the similarity calculation with only *xor* and *popcount* (find detailed derivation from the original paper of [36]).

Until now, we have introduced the process of recurrent binarization when the loop is set as 1, *i.e.*, repeat once. In real-world applications, the loop can be customized to trade off accuracy and efficiency. Formally, the whole process of recurrent binarization with $u \geq 1$ loops can be defined as

$$\text{base binarization: } b_0 = \rho(\boldsymbol{W}_0(f)),$$

$$\text{residual binarization loops: } \begin{cases} \hat{f}_{u-1} = \|\boldsymbol{R}_{u-1}(b_{u-1})\|, \\ r_{u-1} = \rho(\boldsymbol{W}_u(f - \hat{f}_{u-1})), \\ b_u = b_{u-1} + 2^{-u}\boldsymbol{r}_{u-1}. \end{cases}$$

The recurrent binary embedding b_u is the output of the binarization process in Eq. (2), *i.e.*, $b_u = \varphi(f)$ given $f = \phi(q)$ or $f = \phi(d_k)$. Given the output dimension of $\boldsymbol{W}$ as m, the overall bits of b_u can be calculated as $m \times (u + 1)$.

3.2.2 Task-agnostic training. As shown in Eq. (2), the backbone network ϕ and the binarization module φ are commonly jointly optimized in an end-to-end manner in previous learning to hash methods [15]. Though feasible, the training is not efficient given the heavy backbone network for accurate representation learning, and task-dependent as the raw data (*e.g.*, text, images) must be accessed for training end-to-end, rendering an inflexible solution, especially for the data-sensitive applications. To tackle the challenge, we introduce a universal training solution that only requires the float vectors as input, *i.e.*, extracting the embeddings using off-the-shelf backbone networks ϕ. The binarization module φ is therefore trained alone in a task-agnostic and modality-agnostic manner. The objective function of such embedding-to-embedding training can be formulated as

$$\arg\min_{\varphi} \quad \mathcal{L}(\mathcal{F}; \varphi), \tag{3}$$

where $\mathcal{F}$ is the set of all float vectors for training.

Given the great success of contrastive loss [11] in representation learning research, we adopt an NCE-form contrastive objective to regularize the binarization module,

$$\mathcal{L}(\mathcal{F}; \varphi) = \frac{1}{|\mathcal{F}|} \sum_{f \in \mathcal{F}} -\log \frac{\exp\langle \varphi(f), \varphi(k_+)\rangle}{\sum_{k \in \mathcal{B}} \exp\langle \varphi(f), \varphi(k)\rangle}, \tag{4}$$

where k is the float features within the same batch $\mathcal{B}$ as the anchor f. k_+ is the positive sample constructed from another augmented view of an image or the query-document pair from the web. $\langle \cdot, \cdot \rangle$ is cosine similarity between recurrent binary embeddings. Besides the positive pairs collected by manual annotations or user behaviors, we employ hard negative mining to further improve the discriminativeness of the learned binary embeddings.

Inspired by He et al. [19], we enable efficient global hard negative mining by maintaining a queue $\boldsymbol{Q} \in \mathbb{R}^{L \times m}$ of negative sample embeddings. Specifically, we extend the mini-batches with a fix-length (*i.e.*, L) queue (about 16× larger than the mini-batch) and mine hard samples in the queue on the fly. At each training step, the binary embeddings of the current mini-batch are added to the queue, and the oldest mini-batch in the queue is removed if the maximum capacity is reached. Note that we perform momentum updates of the binarization module to encode embeddings for the queue in order to keep latent consistency among different batches, following the practice in [19]. We select the top-k hardest negative samples in the queue for contrastive objectives in Eq. (4), *i.e.*, the samples that receive the highest similarity scores with the anchor feature. Therefore, the set of training samples $\mathcal{B}$ in Eq. (4) becomes

$$\mathcal{B} = \{k_+, \kappa(\boldsymbol{Q})\}, \tag{5}$$

where $\kappa(\boldsymbol{Q})$ denotes the operation for selecting top-k hardest negative samples from $\boldsymbol{Q}$.

Once φ is learned, the recurrent binary embeddings for queries and documents can be produced in an efficient embedding-to-embedding paradigm. Both training and deployment processes are task-agnostic since only full-precision embeddings are needed as input, which enables universal embedding binarization across all the modalities and tasks.

3.2.3 Backward-compatible training. As illustrated in Figure 1, the retrieval stage needs to process billions or trillions of documents. The huge scale of data poses challenges in embedding upgrades since all the index embeddings need to be re-extracted before the deployment of a new model. Such a process is quite time-consuming and computationally expensive. In this paper, we for the first time investigate the potential of backward-compatible learning [37] with binary embeddings. To be specific, compatible learning requires

the embeddings encoded by the old model and the new model to be interchangeable in a consistent latent space. Embedding model upgrades with backward compatibility can deploy the new model immediately without refreshing the index, that is, the new query embeddings can be directly compared with the old document embeddings. The upgrading objective can be formulated as

$$\mathcal{S}_{\text{BEBR-BC}}(q_{\text{new}}, d^{+}_{\text{old}}) \geq \mathcal{S}_{\text{BEBR}}(q_{\text{old}}, d^{+}_{\text{old}}), \tag{6}$$

$$\mathcal{S}_{\text{BEBR-BC}}(q_{\text{new}}, d^{-}_{\text{old}}) \leq \mathcal{S}_{\text{BEBR}}(q_{\text{old}}, d^{-}_{\text{old}}), \tag{7}$$

where d^{+} denotes relevant documents to user query q, and d^{-} denotes the irrelevant ones. $\mathcal{S}_{\text{BEBR-BC}}(\cdot,\cdot)$ calculates the similarity between the new binary embedding of query and old binary embedding of the document, which is formulated as:

$$\mathcal{S}_{\text{BEBR-BC}}(q, d_k) = \mathcal{D}\left(\tilde{\phi} \circ \varphi_{\text{new}}(q), \phi \circ \varphi_{\text{old}}(d_k)\right) \quad \forall k \in \{1, \cdots, n\}, \tag{8}$$

where $\varphi_{\text{new}}(\cdot)$ is the new version of the recurrent binary transformation module and $\varphi_{\text{old}}(\cdot)$ is the old one, $\tilde{\phi}$ denotes a new or identical float backbone model determined by specific applications. BEBR-BC stands for backward compatible BEBR system.

The training objective can be formulated as

$$\arg\min_{\varphi_{\text{new}}} \ \mathcal{L}(\mathcal{F}; \varphi_{\text{new}}) + \mathcal{L}_{\text{BC}}(\mathcal{F}; \varphi_{\text{new}}, \varphi_{\text{old}}), \tag{9}$$

where $\mathcal{L}$ is the same as Eq. (4), and $\mathcal{L}_{\text{BC}}$ is also in the form of an NCE loss but across old and new models, *i.e.*,

$$\mathcal{L}_{\text{BC}}(\mathcal{F}; \varphi_{\text{new}}, \varphi_{\text{old}}) = \frac{1}{|\mathcal{F}|} \sum_{f \in \mathcal{F}} -\log \frac{\exp\langle \varphi_{\text{new}}(\tilde{f}), \varphi_{\text{old}}(k_+)\rangle}{\sum_{k \in \mathcal{B}} \exp\langle \varphi_{\text{new}}(\tilde{f}), \varphi_{\text{old}}(k)\rangle}. \tag{10}$$

$\tilde{f}$ is encoded by $\tilde{\phi}(\cdot)$. φ_{new} is optimized individually with the other parametric modules fixed. $\mathcal{L}$ maintains self-discrimination of the new binarization model while $\mathcal{L}_{\text{BC}}$ regularizes the cross-model compatibility. Queue-based hard mining is also applied for $\mathcal{L}_{\text{BC}}$.

3.3 Deployment

3.3.1 Dot product of recurrent binary embeddings: A revisit. In Shan et al. [36], the cos similarity of recurrent binary embedding is decomposed into the dot product of hash codes as in Eq. (11), where the subscript q and d denote the query and documentation. Thus, the calculation of hash codes can be implemented efficiently with the bit-wise operation as in Eq. (12), where x, y, are binary vectors in $\{1, -1\}^m$, $popc$, $\wedge$ and $>>$ are the population count, XOR, and logical right shift operations.

$$\mathcal{D}(b^q_u, b^d_u) \propto \frac{1}{||b^d||}(b^q_0 \cdot b^d_0 + \sum_{j=0}^{u-1}\sum_{i=0}^{u-1}(\frac{1}{2})^{j+i+2} r^q_j \cdot r^d_i + \sum_{j=0}^{u-1}(\frac{1}{2})^{j+1} b^q_0 \cdot r^d_j + \sum_{i=0}^{u-1}(\frac{1}{2})^{i+1} b^d_0 \cdot r^q_i) \tag{11}$$

$$x \cdot y = (popc(x \wedge y) >> 1) + m \tag{12}$$

Although the bit-wise operation is fast with population count, the computation complexity grows rapidly with the increase of u. Hence, it relies on GPU to offer high performance, and an optimized k-NN selection algorithm is developed.

3.3.2 Symmetric distance calculation (SDC). Unfortunately, the GPU-Enabled NN search algorithm limits its usefulness and applicability in practical cases. In this paper, we develop a Symmetric Distance Calculation (SDC) of recurrent binary embedding around the CPU platform, which is applicable to most scenarios. Specifically, SDC allows computing the distance between the uncompressed recurrent binary features. It relies on SIMD in-register shuffle operation to provide a high-performance calculation procedure, which can be combined with inverted indexes. For simplicity's sake, we explain the SDC uses 128-bit SIMD in the following content.

Similar to [2],[3], and [7], SIMD registers and in-register shuffles are used to store lookup tables and perform lookups. However, these methods use sub-quantizers to obtain different centroids without normalization during calculation. Therefore, algorithmic changes are required to obtain the fixed centroids and magnitude of embeddings for normalization. More specifically, SDC relies on 4-bit code as a basic unit and uses 8-bit integers for storing lookup tables. The resulting lookup tables comprise 16 8-bits integers (128 bits). Once lookup tables are stored in SIMD registers, in-register shuffles can perform 16 lookups in 1 cycle, enabling large performance gains.

Memory layout. By setting $u \in \{1, 3\}$, we first generate the recurrent binary vectors and organize an inverted list of features with the standard memory layout. As shown in the upper of Figure 4. x_i is the 4-bit code where $x \in \{a, b, ..., p\}$, and x_{norm} is the quantized magnitude value of the vector appended at the end. Notably, for $u = 1$, the x_i represents two adjacent dimensions of the feature. To efficiently shuffle lookup tables, the standard memory layout of inverted lists needs to be transposed because the transposed data are contiguous in memory, and the SIMD register can be loaded in a single memory read. This transition process is performed offline and does not influence the search speed.

Lookup tables. As mentioned early, the centroids in SDC are fixed with the setting of u, and it can uncompressed represent the recurrent binary vectors. When $u = 3$, the distance table in 128-bit registers can be reconstructed directly because the centroids of SDC are presented as 4-bit integers, and the inner product distance range is 8-bit integers. When $u = 1$, we use two adjacent dimensions of recurrent binary vector to form 4-bit code, and the distance can be calculated by adding the inner products result of two 2-bit respectively.

SIMD computation. As the lookup tables and inverted list are prepared, each inverted list is scanned block by block. We depicted this process in Figure 4. First, the index codes are packed as 8-bit in each cell of 128-bit registers, and we unpacked the subcodes using shifts and masks. For each set of subcodes, the partial distances are yielded using lookup implementation through a combination of shuffle and blends. This process is repeated $um/4$ times, and the distances are obtained by summing each partial distance with saturated arithmetic. Lastly, each distance is normalized by dividing its magnitude value. In practice, we multiply the distance by the reciprocal of the magnitude value since the multiply operation is fast in SIMD.

3.3.3 ANN systems. We deployed a distributed search system based on ANN search algorithms as in Figure 5. At run time, a query embedding is generated on-the-fly by the embedding learning model. Then, the proxy module dispatches the query to the leaf module,

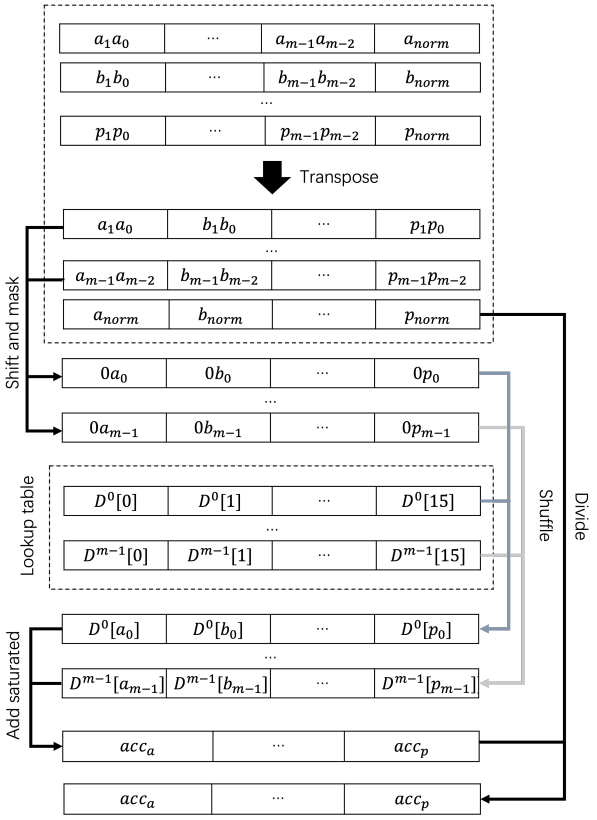

Figure 4: Symmetric distance calculation (SDC) using SIMD calculation.

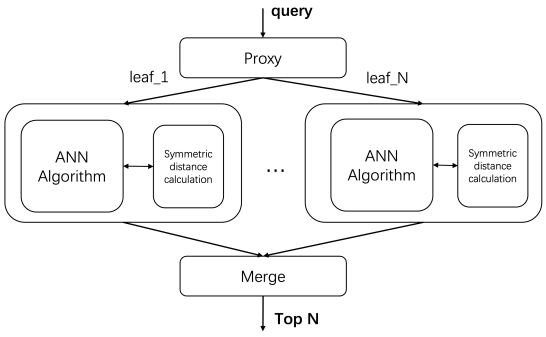

Figure 5: Overview of ANN systems equipped with BEBR.

where the main search process happens. Each leaf module equipped various ANN indexes with symmetric distance calculation since our work is orthogonal to ANN algorithms and compatible with any type of index. Therefore, we can choose different algorithms according to the different requirements of the product. For instance, the inverted index (IVF) has two layers for embedding search, one is the coarse layer quantizes embedding vectors into the coarse cluster typically through the *K-means* algorithm, and the other is the fine-grained layer does the efficient calculation of embedding distances. Both layers can be supported by symmetric distance calculation with recurrent binary embeddings used. Lastly, the result from all leaves will be used to produce the top N result through the selection merge process.

4 EXPERIMENTS

4.1 Implementation Details

The Adam optimizer [24] is employed with an initial learning rate of 0.02. The temperature parameter τ of softmax is set to 0.07. We also adopt gradient clipping when the norm of the gradient exceeds a threshold of 5. When training on a server of 8 Nvidia V100 GPUs, the batch size is set to 4096 for binary representation learning and 128 for compatible learning. The binarization experiments are based on the PyTorch framework. We implemented 256-bit SDC in C++, using compiler intrinsics to access SIMD instructions. g++ version 8.1 are selected and enables SSE, AVX, AVX2, and AVX512. Besides, we use Intel MKL 2018 for BLAS. We carry our experiments on Skylake-based servers, which are Tencent cloud instances, built around Intel(R) Xeon(R) Platinum 8255C CPU @ 2.50GHz.

4.2 Datasets

We evaluate the proposed BEBR on both public and industrial datasets. For the public dataset, we use image and text data from the MS COCO captioning dataset. For the industrial dataset, we use data collected from two applications. One is web search which returns relevant web pages given a user search query. The other one is video copyright detection which identifies duplicated, replicated, and/or slightly modified versions of a given video sequence (query) in a reference video dataset. **Offline datasets:** For web search, we collect search logs of user queries and clicks from *Sogou Search Engine*. After data pre-processing, the training set contains 400 million samples and we use an additional 3 million samples for evaluation. For video copyright detection, we use 8 million images extracted from video sequences to train the model and manually label 30k queries and 600k reference images for validation. Furthermore, we use COCO captioning dataset which contains about 110k training images and 5k validation images. For each image in the training and validation set, five independent human-generated captions are provided. **Online datasets:** We deploy the proposed BEBR in the production environment of the aforementioned two applications. Web search documents are approximately 6 billion in size, covering the most active web pages on the Internet. The size of image embeddings extracted from video sequences in video copyright detection is about 10 billion.

4.3 Evaluation Metrics

Offline evaluation. We use the metric of Recall@k to evaluate the offline performance of the proposed binary-based embedding retrieval method. Specifically, given a query q, its relevant documents $\mathcal{D}^+ = \{d_1^+, \cdots, d_N^+\}$, and the top-k candidates returned by a model as the retrieval set $\hat{\mathcal{D}} = \{d_1, \cdots, d_k\}$. $k \gg N$ in practice. Recall@k is defined as:

$$\text{Recall@}k = \frac{|\mathcal{D}^+ \cap \hat{\mathcal{D}}|}{N} \tag{13}$$

Online Evaluation We use different metrics to evaluate the effectiveness of our proposed BEBR system. For web search, we adopt the metrics of click-through rate (CTR) and query rewrite rate (QRR)

Table 1: Retrieval performance of different embedding forms on MS COCO caption dataset.

Embedding	Bits	Recall@1	Recall@5	Recall@10
hash	1024	0.348	0.632	0.730
ours	1024	**0.360**	**0.646**	**0.751**
float	16384	0.361	0.649	0.744

Table 2: Retrieval performance of different embedding forms on industrial dataset collected from web search and video copyright detection applications.

Embedding	Web search	video copyright detection
	Recall@10	Recall@20
hash	0.819	0.688
ours	**0.853**	**0.727**
float	0.856	0.734

which are believed to be good indicators of search satisfaction. For video copyright detection, we conduct the human evaluation for the performance of retrieved documents, Specifically, we ask human evaluators to label the relevance of results from the BEBR system and baseline system. Apart from precision, we report the ratio between the number of copied videos (positive results) and the number of traffic (denoted as detection ratio) to analyze the model's effect on the entire system. A higher detection ratio indicates better performance. Furthermore, to evaluate the efficiency of the BEBR system, We calculate queries per second (QPS) by measuring the amount of search traffic the retrieval stage receives in one second. We also investigate the memory consumption of the search index built in the retrieval stage.

4.4 Offline Evaluation

Effectiveness of recurrent binary embedding. We investigate the effectiveness of recurrent binary embeddings on both public (academic) and private (industrial) benchmarks. As demonstrated in Tables. 1&2, we compare with the baseline hash [40] (1 bit per dimension) and the oracle float (full precision embedding, 32 bits per dimension).

For the academic dataset, we conduct image-to-text retrieval experiments using the MS COCO caption dataset. Specifically, we employ CLIP [34] model of ResNet101 to produce float embedding for image and text data. The float embedding of size 16384 bits is then compressed into recurrent binary embedding and hash vector with a size of 1024 bits, achieving a 16x compression ratio. As shown in Table 1, recurrent binary embedding surpasses hash embedding and achieves comparable results with float embedding.

For industrial datasets, the vector size of float embeddings in web search and video copyright detection are 8192 and 4096 bits respectively. We adopt the same compression ratio setting of 16x by compressing them into binary embeddings with sizes of 512 and 256 bits respectively. The results are shown in Table 2, we achieve comparable retrieval performance with float embedding in web search and video copyright detection applications and surpass hash embedding by 2.4% and 3.9% respectively.

Table 3: Comparison with alternative options of binary training pipeline. Experiments are conducted on a web search dataset with 400 million training samples. φ denotes recurrent binary model, ϕ and ψ denote encoder model for queries and documents.

Training pipeline	Recall@10	Training time
end-to-end	0.855	125 GPU hours
train φ only (fixed ϕ, ψ)	0.853	125 GPU hours
embedding-to-embedding	0.853	**11** GPU hours

Table 4: Comparison with alternative options of backward-compatible training. (φ_{new}, φ_{old}) denotes using binary embeddings produced by the new binary model to search binary embeddings produced by the old binary model.

Learning strategy	Comparison pair	Recall@20
baseline	(φ_{old}, φ_{old})	0.727
normal bct	(φ_{new}, φ_{old})	0.765
two-stage bct	(φ_{new}, φ_{old})	0.783
ours	(φ_{new}, φ_{old})	**0.801**

Comparison with alternative options of binary training. To investigate the effectiveness and efficiency of our task-agnostic binary training pipeline, we compare it with two alternative options of binary training pipeline. One is end-to-end training where the recurrent binarization module is optimized end-to-end with the backbone network (ψ and ϕ). The other one adopts a similar pipeline to the end-to-end training but with parameters in ψ and ϕ fixed. The fixed ψ and ϕ models help improve the retrieval performance by providing data augmentations. Results are shown in Table 3. Our proposed task-agnostic embedding-to-embedding training pipeline achieves comparable performance with the other two end-to-end training pipelines while reducing training time by 91.2%.

Comparison with alternative options of backward compatible training. We investigate the effectiveness of our proposed backward-compatible training pipeline by comparing with two other pipelines. We denote the first alternative pipeline as normal bct where backward compatible training is conducted between new query encoder ψ_{new} and old document encoder ϕ_{old}. During deployment, backward-compatible binary embeddings are obtained by mapping full precision queries and document embeddings into a common space using the old binary model φ_{old}. The second alternative pipeline (denote as two-stage bct) contains a two-stage training process where the first stage learns backward compatible full precision embeddings, and the second stage learns backward compatible recurrent binary embeddings based on the compatible output of the first stage.

All experiments of compatible learning are conducted on an offline dataset collected from video copyright detection applications. Results are shown in Table 4. All three learning strategies achieve solid backward compatibility by surpassing the baseline where indexing is conducted between the old version of recurrent binary embeddings, indicating the applicability of backward-compatible training in binary embeddings. Among them, our proposed learning

Table 5: Latency for exhaustive search on CPU platform in video copyright detection application.

Embedding	Index type	Bits	Search(s)↓	QPS↑
hash code	bitwise	256	0.0024	414
ours ($u = 1$)	bitwise	256	0.0032	312
ours ($u = 1$)	SDC	256	**0.0020**	**480**
ours ($u = 3$)	bitwise	256	0.0054	185
ours ($u = 3$)	SDC	256	**0.0020**	**480**
float	flat	4096	0.0510	19

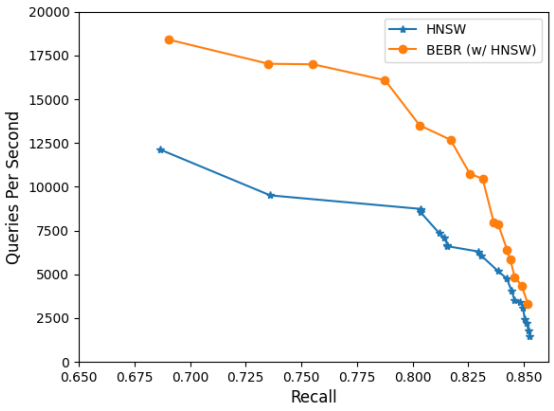

Figure 6: Comparison of retrieval efficiency before and after the deployment of BEBR. Experiments are conducted on an offline dataset collected from web search.

paradigm learns better backward compatibility which outperforms normal bct and two-stage bct by 3.6% and 1.8%.

Search latency on CPU platform. We implement the standard distance calculation between recurrent binary embeddings on the CPU by using the *popcount* operation and carry exhaustive search experiment on the offline video copyright detection dataset with recall@20. The experiment loop is, by default, run on a single CPU in a single thread. The comparison between bit-wise operation and SDC results is shown in Table 5. We observe the bit-wise-based method continues to decrease in QPS with an increase of u, and the SDC is almost 2 times faster than the bit-wise operation at $u = 3$. Notably, SDC is slightly faster than hash code since the shuffle instructions used in SDC are faster than *popc*.

Integration of BEBR into ANN algorithms. Besides the exhaustive search experiments in Table 5, we also conduct experiments that integrate BEBR into ANN algorithms. Specifically, we equip the HNSW algorithm with the symmetric distance calculation component and leverage recurrent binary embedding for search. Results are illustrated in Figure 6. After deploying BEBR, HNSW achieves significant improvements in retrieval efficiency.

4.5 Online A/B Test

We deploy the binary embedding-based retrieval system to web search and video copyright detection applications in Tencent and compare them to strong baselines which utilize full precision embedding for retrieval. Note that we substitute full precision embedding with recurrent binary embedding only in the retrieval stage. The subsequent re-rank stages are identical for both settings. Here, we would like to focus on the balance of performance and efficiency, where resource usage is recorded.

Table 6: Online A/B tests of BEBR in web search.

CTR	QRR	Memory usage	QPS
-0.02%	-0.07%	**-73.91%**	**+90%**

Table 7: Online A/B tests of BEBR in video copyright detection.

Precision	Detection ratio	Memory usage	QPS
-0.13%	-0.21%	**-89.65%**	**+72%**

The live experiment is conducted over 30% of the service traffic during one week. Table 6 and Table 7 show the great benefits of resource and efficiency while retaining performance at the system level. Specifically, BEBR conserves 73.91% memory usage and increase the QPS of retrieval by 90%, while CTR and QRR of web search application decrease slightly by 0.02% and 0.07% respectively. In video copyright detection, memory usage is reduced by 89.65%, and QPS is increased by 72%, while the precision and detection ratio decreases slightly by 0.13% and 0.21%. The improvements in retrieval efficiency and storage consumption lead to overall cost reduction. After deploying BEBR, the overall costs of retrieval in web search and video copyright detection are reduced by 55% and 31% respectively.

5 CONCLUSION

The paper presents binary embedding-based retrieval (BEBR) to improve retrieval efficiency and reduce storage consumption while retaining retrieval performance in Tencent products. Specifically, we 1) compress full-precision embedding into recurrent binary embedding using a lightweight transformation model; 2) adopt a new task-agnostic embedding-to-embedding strategy to enable efficient training and deployment of binary embeddings; 3) investigate backward-compatible training in binary embeddings to enable refresh-free embedding model upgrades; 4) propose symmetric distance calculation equipped with ANN algorithms to form an efficient index system. BEBR has been successfully deployed into Tencent's products, including web search (Sogou), QQ, and Tencent Video. We hope our work can well inspire the community to effectively deliver research achievements into real-world applications.

ACKNOWLEDGMENTS

We sincerely appreciate all the colleagues in the project of BEBR development, including but not limited to Yang Li, Qiugen Xiao, Dezhang Yuan, Yang Fang, Chen Xu, and Xiaohu Qie, for their valuable discussions, efforts, and support. We thank all the reviewers and chairs for their time and constructive comments.

REFERENCES

[1] Abien Fred Agarap. 2018. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375* (2018).

[2] Fabien André, Anne-Marie Kermarrec, and Nicolas Le Scouarnec. 2017. Accelerated nearest neighbor search with quick adc. In *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*. 159–166.

[3] Fabien André, Anne-Marie Kermarrec, and Nicolas Le Scouarnec. 2019. Quicker adc: Unlocking the hidden potential of product quantization with simd. *IEEE transactions on pattern analysis and machine intelligence* (2019), 1666–1677.

[4] Fabien André, Anne-Marie Kermarrec, and Nicolas Le Scouarnec. 2019. Derived codebooks for high-accuracy nearest neighbor search. *arXiv preprint arXiv:1905.06900* (2019).

[5] Artem Babenko and Victor Lempitsky. 2014. Additive quantization for extreme vector compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 931–938.

[6] Artem Babenko and Victor Lempitsky. 2015. Tree quantization for large-scale similarity search and classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4240–4248.

[7] Davis W Blalock and John V Guttag. 2017. Bolt: Accelerated data mining with fast vector compression. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 727–735.

[8] Mateusz Budnik and Yannis Avrithis. 2021. Asymmetric metric learning for knowledge transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8228–8238.

[9] Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Philip S Yu. 2017. Hashnet: Deep learning to hash by continuation. In *Proceedings of the IEEE international conference on computer vision*. 5608–5617.

[10] Ken Chen, Yichao Wu, Haoyu Qin, Ding Liang, Xuebo Liu, and Junjie Yan. 2019. R3 adversarial network for cross model face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9868–9876.

[11] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*. PMLR, 1597–1607.

[12] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2016. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830* (2016).

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[14] Rahul Duggal, Hao Zhou, Shuo Yang, Yuanjun Xiong, Wei Xia, Zhuowen Tu, and Stefano Soatto. 2021. Compatibility-aware heterogeneous visual search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10723–10732.

[15] Lixin Fan, Kam Woh Ng, Ce Ju, Tianyu Zhang, and Chee Seng Chan. 2020. Deep Polarized Network for Supervised Learning of Accurate Binary Hashing Codes.. In *IJCAI*. 825–831.

[16] Cong Fu, Chao Xiang, Changxu Wang, and Deng Cai. 2017. Fast approximate nearest neighbor search with the navigating spreading-out graph. *arXiv preprint arXiv:1707.00143* (2017).

[17] Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. 2013. Optimized product quantization. *IEEE transactions on pattern analysis and machine intelligence* 36, 4 (2013), 744–755.

[18] Jiafeng Guo, Yinqiong Cai, Yixing Fan, Fei Sun, Ruqing Zhang, and Xueqi Cheng. 2022. Semantic models for the first-stage retrieval: A comprehensive review. *ACM Transactions on Information Systems (TOIS)* 40, 4 (2022), 1–42.

[19] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 9729–9738.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[21] Weihua Hu, Rajas Bansal, Kaidi Cao, Nikhil Rao, Karthik Subbian, and Jure Leskovec. 2022. Learning Backward Compatible Embeddings. *arXiv preprint arXiv:2206.03040* (2022).

[22] Jui-Ting Huang, Ashish Sharma, Shuying Sun, Li Xia, David Zhang, Philip Pronin, Janani Padmanabhan, Giuseppe Ottaviano, and Linjun Yang. 2020. Embedding-based retrieval in facebook search. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2553–2561.

[23] Herve Jegou, Matthijs Douze, and Cordelia Schmid. 2010. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence* 33, 1 (2010), 117–128.

[24] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[25] Sen Li, Fuyu Lv, Taiwei Jin, Guli Lin, Keping Yang, Xiaoyi Zeng, Xiao-Ming Wu, and Qianli Ma. 2021. Embedding-based product retrieval in taobao search. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 3181–3189.

[26] Wu-Jun Li, Sheng Wang, and Wang-Cheng Kang. 2015. Feature learning based deep supervised hashing with pairwise labels. *arXiv preprint arXiv:1511.03855* (2015).

[27] Yury Malkov, Alexander Ponomarenko, Andrey Logvinov, and Vladimir Krylov. 2012. Scalable distributed algorithm for approximate nearest neighbor search problem in high dimensional general metric spaces. In *Similarity Search and Applications: 5th International Conference, SISAP 2012, Toronto, ON, Canada, August 9-10, 2012. Proceedings 5*. Springer, 132–147.

[28] Yu A Malkov and Dmitry A Yashunin. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence* 42, 4 (2018), 824–836.

[29] Qiang Meng, Chixiang Zhang, Xiaoqiang Xu, and Feng Zhou. 2021. Learning compatible embeddings. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 9939–9948.

[30] Alistair Moffat and Justin Zobel. 1996. Self-indexing inverted files for fast text retrieval. *ACM Transactions on Information Systems (TOIS)* 14, 4 (1996), 349–379.

[31] Priyanka Nigam, Yiwei Song, Vijai Mohan, Vihan Lakshman, Weitian Ding, Ankit Shingavi, Choon Hui Teo, Hao Gu, and Bing Yin. 2019. Semantic product search. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2876–2885.

[32] Mohammad Norouzi and David J Fleet. 2013. Cartesian k-means. In *Proceedings of the IEEE Conference on computer Vision and Pattern Recognition*. 3017–3024.

[33] Zijing Ou, Qinliang Su, Jianxing Yu, Ruihui Zhao, Yefeng Zheng, and Bang Liu. 2021. Refining BERT Embeddings for Document Hashing via Mutual Information Maximization. *arXiv preprint arXiv:2109.02867* (2021).

[34] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*. PMLR, 8748–8763.

[35] Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval* 3, 4 (2009), 333–389.

[36] Ying Shan, Jian Jiao, Jie Zhu, and JC Mao. 2018. Recurrent binary embedding for gpu-enabled exhaustive retrieval from billion-scale semantic vectors. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2170–2179.

[37] Yantao Shen, Yuanjun Xiong, Wei Xia, and Stefano Soatto. 2020. Towards backward-compatible representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6368–6377.

[38] Shupeng Su, Chao Zhang, Kai Han, and Yonghong Tian. 2018. Greedy hash: Towards fast optimization for accurate hash coding in cnn. *Advances in neural information processing systems* 31 (2018).

[39] Chien-Yi Wang, Ya-Liang Chang, Shang-Ta Yang, Dong Chen, and Shang-Hong Lai. 2020. Unified representation learning for cross model compatibility. *arXiv preprint arXiv:2008.04821* (2020).

[40] Xiaofang Wang, Yi Shi, and Kris M Kitani. 2017. Deep supervised hashing with triplet labels. In *Computer Vision–ACCV 2016: 13th Asian Conference on Computer Vision, Taipei, Taiwan, November 20-24, 2016, Revised Selected Papers, Part I 13*. Springer, 70–84.

[41] Tao Wu, Ellie Ka-In Chio, Heng-Tze Cheng, Yu Du, Steffen Rendle, Dima Kuzmin, Ritesh Agarwal, Li Zhang, John Anderson, Sarvjeet Singh, et al. 2020. Zero-shot heterogeneous transfer learning from recommender systems to cold-start search retrieval. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2821–2828.

[42] Binjie Zhang, Yixiao Ge, Yantao Shen, Yu Li, Chun Yuan, Xuyuan Xu, Yexin Wang, and Ying Shan. 2022. Hot-Refresh Model Upgrades with Regression-Alleviating Compatible Training in Image Retrieval. *arXiv preprint arXiv:2201.09724* (2022).

[43] Ting Zhang, Chao Du, and Jingdong Wang. 2014. Composite quantization for approximate nearest neighbor search. In *International Conference on Machine Learning*. PMLR, 838–846.

[44] Chang Zhou, Lai-Man Po, Wilson YF Yuen, Kwok Wai Cheung, Xuyuan Xu, Kin Wai Lau, Yuzhi Zhao, and Mengyang Liu. 2019. Angular deep supervised hashing for image retrieval. *IEEE Access* 7 (2019), 127521–127532.

A APPENDIX

In this section, we provide detailed information about retrieval-based applications at Tencent, to facilitate a better understanding of the contributions of the proposed BEBR.

A.1 Products in Tencent PCG

Figure 7: Illustration of partial products at Tencent PCG.

Tencent is a world-leading internet and technology company that develops innovative products and services to improve the quality of life of people around the world. Our group, Tencent PCG (Platform and Content Group), is a large business group running Tencent's social media, information feed, search, and content platforms, taking Sogou, Tencent Video, and QQ as examples (as demonstrated in Figure 7). Large-scale index-based information search (*e.g.*, web/video search) and copyright detection are two fundamental and essential AI capabilities for these products.

A.2 Information Search

Almost all products at Tencent require searching for useful information that meets the needs of users. For example, the search engine product, Sogou[2], helps users to acquire information by providing relevant videos, news, and queries under different search sessions. Video feeds product returns personalized results of relevant videos to improve user experience. Large scale EBR system has been developed as infrastructure at Tencent to provide support to search-related applications. However, huge costs in computation and storage as well as search efficiency still remain intractable due to super-scale index and high concurrent queries.

Fortunately, after the deployment of BEBR, up to 73.91% reduction in both memory and hard disk consumption has been achieved with almost no accuracy loss at the system level, taking one more step towards Green AI. Furthermore, the search efficiency has also been greatly improved with the help of symmetric distance calculation. The released resources support further development of emerging functions and sustainable development of the products.

[2]http://sogou.com/

A.3 Video Copyright Detection

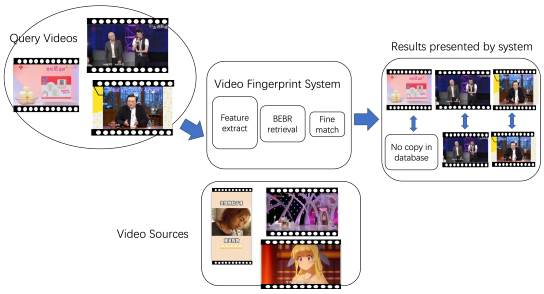

Figure 8: Illustration of video fingerprint system used in video copyright detection applications.

Tencent Video (also known as WeTV outside of china) is one of China's largest online video platforms. As of March 2022, Tencent Video has over 1.268 billion monthly mobile active users and 123 million VIP subscribers. Therefore, there are huge traffic videos, and a large number of videos are uploaded on Tencent video every day. This poses great problems for the company in identifying the illicit versions of the original data.

The *video fingerprint* system, which serves copyright detection of Tencent Video, is developed to generate unique and robust identification of the uploaded videos. As shown in Figure 8, the system consists of feature extraction, large-scale retrieval, and fine-grained matching stages, in which the retrieval stage takes almost half of the cost in data computation, memory, and disk storage. Upon the deployment of BEBR, we considerably reduce the cost of the retrieval stage by around 60% and the overall cost by around 31%. In addition, the fast computation of SDC enables a lower response time, alleviating the pressure on the entire system and improving the user experience.

Table 8: Backward compatibility in sequential model update scenarios. v2->v1 denotes using query embedding from version 2 to perform index with document embedding from version 1.

Query->Doc	v1->v1	v2->v1	v3->v1
Recall@20	0.727	0.891	0.859

A.4 More Experimental Results

Sequential backward compatibility. We conduct experiments to investigate backward compatibility in the context of multiple update steps. Specifically, we updated the query embeddings twice to version-2 and version-3 respectively, while keeping the document embeddings unchanged at version-1. Experiments are conducted on Tencent's video copyright detection dataset. As shown in Table 8, the retrieval accuracy keeps improving with the upgrades of query embeddings, demonstrating the effectiveness of backward-compatible training at multiple update steps.

Altogether, backward-compatible training enables index across different versions of embeddings by maintaining the latent consistency of different models during training. Even if the index consists

Table 9: Retrieval results at different number of recurrent times u. The overall bits for the binary embedding are kept at 256.

	$u = 0$	$u = 1$	$u = 2$	$u = 3$
Recall@20	0.692	0.727	0.718	0.716

of embeddings extracted from different versions of models (e.g., v1 and v2), the new v3 model can still be deployed immediately without waiting for any replacement or refresh.

Effects of recurrent time u. We further conducted experiments on video copyright detection to investigate the retrieval accuracy at different values of u. From the results in Table 9, it can be observed that the best performance is achieved when u=1. It is worth noting that the performance of all three different settings of recurrent binary embedding ($u \neq 0$) is superior to that of hash embeddings (u=0), indicating the superiority of recurrent binary embeddings.

Table 10: Retrieval results at different overall bit sizes of recurrent binary embeddings. The experiments are conducted on Tencent's video copyright detection dataset.

overall bits	64	128	256	512	4096(float)
Recall@20	0.571	0.644	0.727	0.730	0.734

Effects of overall bit sizes. Given the optimal $u = 1$, we illustrate the retrieval accuracy at different overall bit sizes by varying output dimension m. The results are shown in Table 10. The retrieval performance began to plateau after 256 bits, suggesting that the optimal latency/recall trade-off is achieved around 256 bits, which is adopted as the default setting in our paper.