# Improving Dual-Encoder Training with Gradient-Based Cached Embedding Refinement

**Anonymous ACL submission**

## Abstract

We present an efficient approach for training dual-encoder models in large-scale retrieval tasks, aimed at reducing computational overhead and improving retrieval performance. Our method introduces a novel combination of similarity-based large negative sampling and direct gradient updates to cached target embeddings. By leveraging a pre-trained encoder to initialize target embeddings and storing them in a buffer, we eliminate the need for frequent recomputation, thereby reducing both computational cost and memory usage. Negative samples are selected from the top-$k$ most similar target embeddings within the batch and across queries, and cached embeddings are updated directly through gradient descent. Additionally, we utilize the Faiss library to manage nearest neighbor search, periodically rebuilding the index to maintain efficiency. Our approach accelerates training and improves retrieval accuracy, especially for quantized index types, providing a scalable solution for large-scale retrieval tasks that balances both computational efficiency and retrieval precision.

## 1 Introduction

Efficient retrieval of relevant information from large document corpora is a critical task in modern retrieval-based systems, underpinning applications like open-domain question answering, search ranking, and fact verification (Kwiatkowski et al., 2019; Xu et al., 2024; Chen et al., 2022). Traditional sparse retrieval methods, such as BM25, have long been employed for document matching but face challenges, particularly with vocabulary mismatch and the reliance on bag-of-words embeddings (Croft et al., 2010).

Recent advancements in dense retrieval aim to overcome these limitations by leveraging continuous vector embeddings learned through deep neural networks, enabling more flexible and powerful

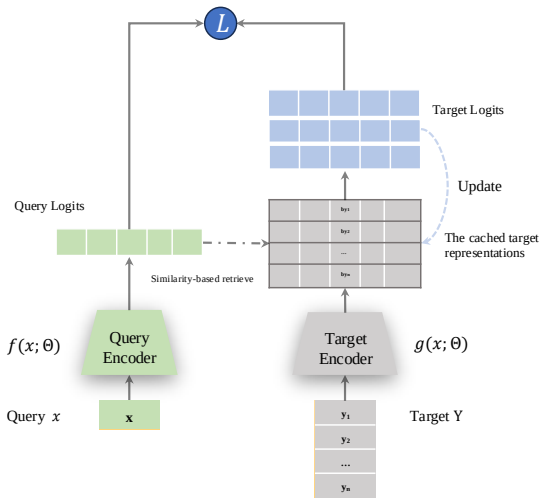


Figure 1: Overview of our approach. We use a pre-trained $g$ to embed all target texts $\mathcal{Y}$. For any given query $x$, the $f$ produces a query embedding. From the cached target embeddings, the top-$k$ most similar negative samples are selected. The target embeddings are directly updated based on the gradients during training.

document matching (Zhao et al., 2024). Among these, dual-encoder models have gained prominence. In this architecture, an input query and candidate document are encoded separately by neural networks, and their relevance is determined by the inner product of their respective embeddings (Karpukhin et al., 2020). However, training these models presents a challenge: directly computing softmax logits over all possible target documents is computationally infeasible. As a result, approximations, such as truncated softmax and negative sampling, are employed to reduce the computational burden, though these methods still suffer from inefficiencies due to the staleness of cached target embeddings (Reddi et al., 2019; Lindgren et al., 2021). Additionally, constructing effective negative instances during training remains a significant challenge, especially in the first stage of retrieval, where models must distinguish relevant

from irrelevant documents (Karpukhin et al., 2020).

In this paper, we propose a novel method to enhance dual-encoder training for large-scale retrieval tasks. Our approach combines similarity-based large negative sampling with direct gradient updates to cached target embeddings. By initializing target embeddings with a pre-trained encoder and storing them in a buffer, we eliminate the need for frequent recomputation, thereby reducing both computational overhead and memory usage. We use truncated softmax for efficient gradient computation and select negative samples from the top-$k$ most similar targets within the batch and across queries, where $k$ can be large to build a more comprehensive negative sample set. The cached target embeddings are then updated through gradient descent. We further integrate the Faiss library for efficient nearest neighbor search and periodically rebuild the index to maintain retrieval accuracy.

Our key contributions are: 1. Efficient target embedding cache with direct gradient updates, eliminating the need for costly recomputations and ensuring up-to-date embeddings. 2. The introduction of similarity-based large negative sampling, combined with in-batch negative samples, to better approximate the full softmax distribution and improve the training efficiency. 3. Experimental results showing significant improvements in retrieval accuracy, particularly with quantized indices, achieving faster retrieval and reduced index size for large-scale tasks.

## 2 Related Work

**Sparse and Dense Retrieval Methods** Traditional retrieval methods, such as TF-IDF and BM25 (Robertson et al., 2009), rely on lexical overlap and are limited in capturing semantic relationships. Dense Passage Retrieval (DPR) (Karpukhin et al., 2020) improves upon these methods by learning dense vector embeddings for both queries and passages, which enhances retrieval accuracy (Zhan et al., 2021; Zhao et al., 2024; Luan et al., 2021). However, training dense models on large corpora remains computationally expensive (Arabzadeh et al., 2021; Monath et al., 2024).

**Negative Sampling** Negative sampling is vital for training dense retrieval models. Traditional methods, such as random and batch-based sampling, can limit diversity. Approaches like using BM25 for "hard" negatives (Karpukhin et al., 2020) improve sample quality but still face limitations. Approximate Nearest-Neighbor Contrastive Learning (ANCE) (Xiong et al., 2020) enhances this by selecting hard negatives based on similarity, improving the quality of negative samples. Dynamic Indexes (Monath et al., 2023) use tree structures for more efficient negative mining, while methods like TriSampler (Yang et al., 2024) further improve performance by providing more informative negatives.

**Target Embedding Caching and Indexing** During training, methods like ANCE (Xiong et al., 2020) and Corrector Networks (Monath et al., 2024) often build a cache of target embeddings to avoid recalculating embeddings for every query. In passage retrieval tasks (Sachan et al., 2022), caching target embeddings is also used to accelerate retrieval by enabling fast lookup during query processing. Several indexing methods are employed to balance retrieval speed and memory usage, including Exact Search for L2 distances (Norouzi et al., 2013), Hierarchical Navigable Small World (HNSW) graphs (Lin and Zhao, 2019), and Inverted File Indexes with post-verification (Kukreja et al., 2023). Product Quantization (PQ) (Thakur et al., 2022) is also commonly used to compress the index size at the expense of retrieval accuracy, offering a scalable solution for large datasets with limited memory.

## 3 Preliminaries

This section introduces the foundational concepts of dual-encoder architectures in dense retrieval models, the softmax function used for ranking relevance, and techniques like truncated softmax and top-k sampling that make training efficient at scale.

**Dual-Encoder Architecture** In dense retrieval models, a dual-encoder architecture is commonly used to compute the unnormalized logits, $s_{x,y}$, by factorizing the embeddings of both the query and the target. Specifically, each input (the query $x$ and the target $y$) is encoded using deep neural networks into $D$-dimensional embeddings. The query $x$ is encoded by a function $f(x;\Theta)$, and the target $y$ is encoded by a function $g(y;\Theta)$. The logits are then computed as the dot product between the query and target embeddings:

$$s_{x,y} = \langle f(x;\Theta), g(y;\Theta) \rangle. \quad (1)$$

This architecture is effective in learning semantic embeddings for both queries and targets, facil-

itating retrieval tasks by ranking the relevance of candidate passages based on the similarity between their embeddings.

**Softmax Function** Given the dual-encoder framework, we now define a probability distribution over a set of $N$ *targets*, $\mathcal{Y}$, based on the similarity between the query $x$ and the target $y$. The softmax function is commonly used to compute this distribution as follows:

$$P(y|x) = \frac{\exp(\tau s_{x,y})}{Z_x} = \frac{\exp(\tau s_{x,y})}{\sum_{y' \in \mathcal{Y}} \exp(\tau s_{x,y'})}, \quad (2)$$

where $\tau$ is the temperature parameter that controls the smoothness of the distribution, and $s_{x,y}$ represents the unnormalized score or logit of the target $y$ given the query $x$. This formulation is commonly used in retrieval tasks, where $x$ corresponds to a query and the targets, $y$, represent candidate passages (e.g., in the Natural Questions dataset (Kwiatkowski et al., 2019), where $x$ is a question, and the targets are passages from Wikipedia).

The softmax function assigns a probability to each target based on its relevance to the query. This is useful in tasks such as information retrieval, where the goal is to rank passages according to their relevance to a given query.

**Training with Truncated Softmax** In the dual-encoder framework, training typically involves optimizing a task-specific loss function, such as cross-entropy, using gradient descent (Rawat et al., 2019). This process requires computing the softmax distribution over all possible targets, which is computationally expensive due to the large number of potential candidates (often in the millions or billions). The exact computation of the normalizing constant $Z_x$—which requires evaluating the sum over all targets—becomes intractable during training.

To address this challenge, we approximate the softmax function by using a truncated version, $\tilde{P}(y|x)$, which includes only a subset of targets $S(\mathcal{Y}) \subset \mathcal{Y}$:

$$\tilde{P}(y|x) = \frac{\exp(\tau s_{x,y})}{\sum_{y' \in S(\mathcal{Y})} \exp(\tau s_{x,y'})}. \quad (3)$$

By truncating the softmax to a subset $S(\mathcal{Y})$, the computation of the normalization term is significantly reduced, making it feasible to perform efficient optimization of the dual-encoder parameters. While this truncated softmax introduces a bias—since it only considers a subset of targets—the approximation enables tractable training while preserving the model's ability to learn effective semantic embeddings.

**Top-K Sampling Approximations** Efficiently selecting the subset $S(\mathcal{Y})$ for truncated softmax is crucial. Traditional methods like BM25 select top-$k$ candidates based on lexical similarity. For dense retrieval, top-$k$ targets are selected based on embedding similarity, measured using the inner product between query and target embeddings. Advanced methods, such as Gumbel-Max (Lindgren et al., 2021) and large-scale sampling algorithms (Xiong et al., 2020), offer more efficient top-$k$ selection, improving retrieval performance while scaling to large datasets.

## 4 Methodology

In this section, we present an efficient approach to training the dual-encoder model, focusing on two main objectives: enhancing the query encoder's learning process and optimizing the update of target embeddings through cached embeddings. The key idea is to utilize similarity-based sampling to select hard negative samples and directly update the cached target embeddings using gradients, eliminating the need to re-evaluate the target encoder during training. This significantly reduces computational overhead, making it feasible to train large-scale retrieval models. An overview of the overall architecture is shown in Figure 1, highlighting the key components and workflow of the proposed approach.

**Target embedding Initialization** To address the computational challenges of large-scale training, we begin by initializing the target embeddings using a pre-trained target encoder, $g(y; \Theta)$. For each target $y_i$, the corresponding embedding is computed as:

$$b_{y_i} = g(y_i; \Theta), \quad (4)$$

where $b_{y_i} \in \mathcal{R}^D$ represents the $D$-dimensional vector for target $y_i$. These initial embeddings are stored in a buffer, $B \in \mathcal{R}^{\mathcal{Y} \times D}$, which contains the cached embeddings for all targets in the dataset.

The interaction between the query encoder and a cached target embedding is computed as the dot

product:

$$s_{x,b_y} = \langle f(x;\Theta), b_y \rangle, \tag{5}$$

where $b_y$ represents the cached embedding of a target stored in the buffer.

Using these scores, the truncated softmax distribution is expressed as:

$$\tilde{P}(b_y|x) = \frac{\exp(\tau s_{x,b_y})}{\sum_{b_{y'} \in S(B)} \exp(\tau s_{x,b_{y'}})}, \tag{6}$$

where $S(B)$ denotes the subset of cached target embeddings selected from the buffer.

By focusing on a subset of target embeddings in the buffer, we can avoid recalculating target embeddings using the target encoder. This approach drastically reduces memory usage and computation, allowing us to sample a large number of high-similarity negative samples efficiently. Consequently, this strategy accelerates the training process without compromising the model's ability to learn high-quality semantic embeddings.

**Negative Sample Construction** A critical component of this methodology is the construction of negative samples, which are essential for effective training of the dual-encoder model. In large-scale retrieval tasks, the goal is to select negative samples that are highly similar to the query embedding, as these "hard" negatives are crucial for optimizing the training objective.

We adopt a hybrid strategy for constructing the negative sample set, $S_{\text{neg}}(B)$, for each query. This set consists of:

- The top-$k$ most similar target embeddings (excluding the positive sample) retrieved based on the query embedding.
- The top-$k$ most similar target embeddings for all other queries in the same batch.

Thus, for each query, the negative sample set contains $k \times$ batch negative samples. For instance, with a batch size of 32 and $k$ set to 500, the total number of negative samples is 16,000. This is a large number compared to those used in other studies.

This strategy ensures that the negative sample set captures both highly relevant hard negatives and diverse negatives across the batch. To improve the approximation of the softmax distribution, $k$ can be set to a large value (e.g., 500). The complete sample set $S_{\text{neg}}(B)$ for each query is then formed by including the corresponding positive sample $b_y$.

**Cached Target embedding Update** We update only the selected cached target embeddings using gradients, which reduces the overall training time and memory usage.

To further enhance training efficiency, we introduce a batch update mechanism for the cached target embeddings. Let $S_{\text{neg}}(B)$ denote the subset of cached target embeddings selected from the buffer for each query. These embeddings are the ones that will undergo gradient updates during training.

After selecting the top-$k$ negative samples for each query, we compute the gradients of the cross-entropy loss with respect to the cached target embeddings. The loss function, $\mathcal{L}$, based on the truncated softmax, is defined as:

$$\mathcal{L} = -\sum_{x \in \mathcal{B}} \log \tilde{P}(b_y|x), \tag{7}$$

where $\tilde{P}(b_y|x)$ is computed using the truncated softmax equation from earlier. Based on the gradients of the loss with respect to the cached target embeddings $b_y$, the cached target embeddings are updated as follows:

$$b_y \leftarrow b_y - \eta \frac{\partial \mathcal{L}}{\partial b_y}, \tag{8}$$

where $\eta$ is the learning rate. This update is applied only to the target embeddings selected in the batch $S(B)$.

After updating the target embeddings for the batch, the modified entries in the buffer are replaced with their updated values. These updated buffer entries are then used for the next training step. Additionally, the similarity-based sampling process ensures that the top-$k$ negative samples are dynamically selected based on the updated embeddings.

**Faiss Index Construction and Update** To efficiently retrieve the top-$k$ most similar samples during similarity-based sampling, directly computing the inner product similarity scores and sorting them is computationally infeasible. A common practice is to use the Faiss library to construct an index that enables fast retrieval of the top-$k$ most similar samples. The most basic index is the flat index, which stores the full vectors and performs exhaustive search. This index maximizes accuracy and is often used in passage retrieval tasks to report optimal test results. However, it is memory-intensive and slow for large-scale datasets.

To address these limitations, we employ the HNSWFlat index, which is based on the HNSW graph. This index allows for fast approximate nearest neighbor searches but still requires significant memory. To further optimize memory usage and retrieval speed, we combine HNSWFlat with Inverted File with Product Quantization (IVFPQ) index. IVFPQ compresses the index size significantly while maintaining high retrieval speed, making it suitable for large-scale training and retrieval tasks.

As the cached target embeddings are periodically updated, the Faiss index constructed on must also be updated to maintain fast and accurate nearest neighbor searches. To ensure efficient retrieval, we rebuild the Faiss index at fixed intervals, typically every 0.5 epochs, to reflect the updated target embeddings.

**Algorithmic Process** The overall training process, including query encoding, retrieval of negative samples, loss calculation, and periodic updates to the Faiss index, is outlined in Algorithm 1. This procedure ensures the efficient training of the model without the need to repeatedly re-evaluate the target encoder, while also enabling scalable training with large datasets.

**Algorithm 1** Improving Dual-Encoder Training with Gradient-Based Cached Embedding Refinement

**Require:** Pre-trained query encoder $f$, target encoder $g$
**Require:** Target dataset $\mathcal{Y} = \{y_i\}_{i=1}^{|\mathcal{Y}|}$
**Require:** Learning rate $\eta$, batch size, negative sample size $k$, faiss index update interval $\mathcal{C}$
**Ensure:** Optimized query encoder $f$ and updated cached target embeddings $B$
Compute initial target embeddings: $B = \{g(y_i) \mid \forall y_i \in \mathcal{Y}\}$
Construct Faiss index for $B$
**for** each training step **do**
    Sample a batch of queries $\mathcal{B} = \{x_j\}_{j=1}^{\text{batch}}$
    Compute query embeddings: $e_{x_j} = f(x_j)$
    Retrieve top-$k$ hard negatives from $B$ for each query
    Compute loss $\mathcal{L}$
    Update query encoder parameters
    Update $B$ with modified target embeddings
    **if** training step $\mod \mathcal{C} == 0$ **then**
        Rebuild Faiss index with updated $B$
    **end if**
**end for**

## 5 Experiments

In this section, we evaluate the effectiveness and efficiency of our proposed method, Efficient Dual-Encoder Training with Gradient-Based Cached Embedding Updates, on large-scale retrieval tasks. Our primary goals are to demonstrate the following key improvements:

- Enhanced query encoder training through similarity-based large negative sampling.
- Scalable and efficient updates to cached target embeddings through direct gradient-based refinement.

The experiments were conducted on two benchmark datasets: **Natural Questions (NQ)** (Kwiatkowski et al., 2019) and **TriviaQA** (Joshi et al., 2017). The NQ dataset is specifically designed for end-to-end question answering tasks, containing real Google search queries as questions, with corresponding answers identified by human annotators in the form of spans from Wikipedia articles. TriviaQA, on the other hand, consists of trivia questions with corresponding answers that were scraped from the web. The dataset includes over 21 million Wikipedia passages, which serve as the target data for retrieval.

For evaluation, we primarily use **Recall@k (R@k)** as the metric, which quantifies the fraction of queries for which the relevant passage is ranked among the top $k$ results.

For all experiments, the query encoder and target encoder were initialized using pre-trained models from the DPR framework. The pre-trained target encoder was used to embed all target texts, generating an initial buffer $B \in \mathcal{R}^{\mathcal{Y} \times D}$ that stores the target embeddings. We trained the models for 10 epochs with a learning rate of $10^{-5}$ using the Adam optimizer, linear learning rate scheduling with warm-up, and a dropout rate of 0.1. The batch size was set to 32, with a top-k of 500 and a temperature of 8. We conducted the training on a system equipped with two NVIDIA 3090 GPUs (24GB memory each), an Intel(R) Xeon(R) CPU E5-2678 v3 @ 2.50GHz, and 125GB of RAM. The training was limited to 24 hours per session to ensure stability and manage resource usage effectively.

### 5.1 Experimental Results

In this section, we compare the performance of our method with several approaches. First, we

use DPR (Karpukhin et al., 2020) as our baseline method. Next, we consider the ANCE method (Xiong et al., 2020), which shares similarities with our approach. Both methods leverage the DPR framework for initialization. However, the ANCE method asynchronously updates a Faiss index using a stale target encoder, and periodically re-embeds all target embeddings. In addition, we compare with the Corrector Networks (Monath et al., 2024), which proposes a scalable solution to this problem by training a small parametric corrector network that adjusts stale cached target embeddings, enabling an accurate softmax approximation and sampling of high-scoring "hard negatives" in an efficient manner.

Table 1 and Table 2 presents the retrieval performance of the various methods on the NQ and TriviaQA test datasets using the FlatL2 indexing method, evaluating Recall@k for different values of $k$. Our approach consistently outperforms the other methods across multiple top-$k$ retrievals. While the performance of our method is comparable to other approaches, it achieves notable improvements in efficiency. Specifically, our method, leveraging similarity-based large negative sampling and efficient gradient updates to the cached embeddings, significantly reduces computational complexity and training time, completing in just 10 epochs.

While FlatL2 indexing generally provides the highest retrieval precision on test datasets, practical passage retrieval tasks often require a balance between retrieval precision, speed, and storage efficiency. Table 3 compares the index size and query speed of different indexing methods. The Quantized HNSW (QHNSW) index significantly outperforms FlatL2 in both index size and query speed. Specifically, the QHNSW index is 30x smaller in size and achieves 300x faster query retrieval compared to FlatL2.

Figure 2 further illustrates the retrieval performance of different Faiss index types on the NQ dataset. Our method significantly improves the precision of the QHNSW index. In terms of R@1, our approach achieves a 5 percentage point improvement over the baseline. Even with compression, the precision of the QHNSW index closely approximates that of the uncompressed FlatL2 index. Improving the retrieval precision of quantized indices enhances the effectiveness of passage retrieval, particularly in scenarios where storage space is limited.

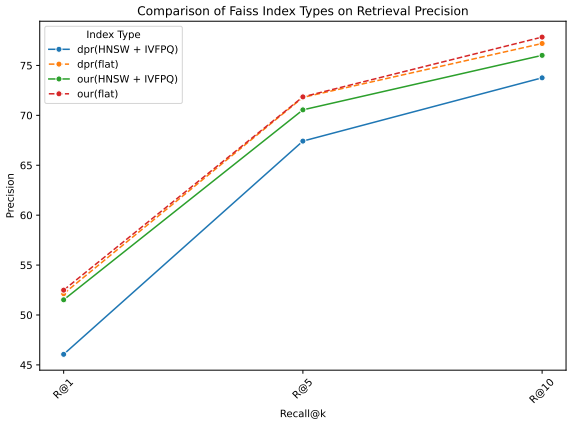


Figure 2: Comparison of retrieval precision using different Faiss index types on the NQ dataset.

### 5.2 Ablation Study on Model Training

To better understand the individual contributions of the components in our proposed method, we conducted an ablation study using the NQ dataset. Specifically, we evaluated the effects of similarity-based large negative sampling and gradient-based updates to cached target embeddings.

We started with a baseline configuration that used the original Dual-encoder model trained with standard settings, without incorporating either similarity-based large negative sampling or gradient-based updates to the target embeddings. The following configurations were then tested:

- **Large-Batch Negatives**: In this configuration, similarity-based large negative sampling was incorporated during query encoder training.
- **Full Model**: This configuration combined both similarity-based large negative sampling and gradient-based updates to the target embeddings, representing the complete approach.

The ablation study results, shown in Table 4, demonstrate the effectiveness of the proposed components. Similarity-based large negative sampling improved retrieval performance across all Recall@k metrics, particularly enhancing the query encoder's ability to differentiate relevant from irrelevant passages. Gradient-based updates to cached target embeddings further boosted performance, especially at higher recall levels, by ensuring the embeddings remain accurate and aligned with the evolving query representations. Combining both strategies in the Full Model led to the best overall

Table 1: Retrieval performance comparison on NQ and TriviaQA test sets (R@k %)

| Dataset | Method | R@1 | R@5 | R@10 | R@20 | R@100 |
|---|---|---|---|---|---|---|
| NQ | DPR | 52.13 | 71.80 | 77.20 | 81.16 | 87.06 |
| | ANCE | - | - | - | 81.9 | 87.5 |
| | Corrector | 50.61 | 71.00 | 77.73 | **82.66** | **88.39** |
| | Ours | **52.49** | **71.85** | **77.84** | 81.91 | 87.40 |

Table 2: Retrieval performance comparison on TriviaQA test sets (R@k %)

| Method | R@20 | R@100 |
|---|---|---|
| DPR | 79.4 | 85.0 |
| ANCE | 80.3 | 85.3 |
| Ours | 80.1 | 85.6 |

Table 3: Index Size and Query Speed Comparison

| Index | Size (GB) | Speed (ms) |
|---|---|---|
| FlatL2 | 61 | 16264 |
| HNSW&IVFPQ | **2.7** | **46** |

performance, outperforming both the Baseline and Large-Batch Negatives configurations. This highlights the complementary benefits of these components in improving retrieval accuracy and efficiency.

Overall, the ablation study confirms that similarity-based large negative sampling enhances the query encoder's discriminative ability, while gradient-based updates ensure that the cached embeddings stay aligned with the evolving query representations. Together, these strategies significantly improve retrieval performance, making the model more efficient for large-scale retrieval tasks.

## 6 Conclusion

In this paper, we introduced an enhanced approach to dual-encoder training by leveraging large-scale similarity-based negative sampling and embedding updates via cached embeddings. Our method enables the truncated softmax to closely approximate the full softmax, improving training efficiency. Direct updates to cached embeddings through backpropagation eliminate the need for continuous target embedding recalculation. Experimental results demonstrate that our approach not only matches the accuracy of models using flatL2 indexing but also outperforms the current best methods on the quantized HNSW index, leading to superior retrieval efficiency and precision in practical passage retrieval tasks.

## 7 Limitations

Our approach, while effective, has some limitations. First, direct updates to cached embeddings via backpropagation require storing the entire set of embeddings in memory, leading to high memory consumption. For example, training on the Wikipedia dataset requires up to 70GB of memory, which may be prohibitive for large datasets. Second, a large number of negative samples must be transferred between memory and GPU, which can create I/O bottlenecks, particularly with large batch sizes and limited system memory. Additionally, our method does not support synchronized updates to the target encoder $g$, making it less suitable for scenarios where the target data changes rapidly. These limitations could be addressed by optimizing memory usage and improving data transfer efficiency, as well as enabling asynchronous updates to the target encoder for dynamic content.

## References

Negar Arabzadeh, Xinyi Yan, and Charles LA Clarke. 2021. Predicting efficiency/effectiveness trade-offs for dense vs. sparse retrieval strategy selection. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 2862–2866.

Jiangui Chen, Ruqing Zhang, Jiafeng Guo, Yixing Fan, and Xueqi Cheng. 2022. Gere: Generative evidence retrieval for fact verification. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2184–2189.

W Bruce Croft, Donald Metzler, and Trevor Strohman. 2010. *Search engines: Information retrieval in practice*, volume 520. Addison-Wesley Reading.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly

Table 4: Ablation study results on the NQ dataset. Retrieval performance is evaluated using the QHNSW index.

| Configuration | R@1 | R@5 | R@10 | R@20 | R@100 |
|---|---|---|---|---|---|
| Baseline | 46.06 | 67.42 | 73.76 | 77.83 | 84.65 |
| Large-Batch Negatives | 49.72 | 68.28 | 74.01 | 78.31 | 84.12 |
| Full Model | **51.52** | **70.55** | **76.01** | **79.81** | **86.15** |


supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.

Sanjay Kukreja, Tarun Kumar, Vishal Bharate, Amit Purohit, Abhijit Dasgupta, and Debashis Guha. 2023. Vector databases and vector embeddings-review. In *2023 International Workshop on Artificial Intelligence and Image Processing (IWAIIP)*, pages 231–236. IEEE.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Peng-Cheng Lin and Wan-Lei Zhao. 2019. A comparative study on hierarchical navigable small world graphs. *Computing Research Repository (CoRR) abs/1904.02077*.

Erik Lindgren, Sashank Reddi, Ruiqi Guo, and Sanjiv Kumar. 2021. Efficient training of retrieval models using negative cache. *Advances in Neural Information Processing Systems*, 34:4134–4146.

Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. 2021. Sparse, dense, and attentional representations for text retrieval. *Transactions of the Association for Computational Linguistics*, 9:329–345.

Nicholas Monath, Will Grathwohl, Michael Boratko, Rob Fergus, Andrew McCallum, and Manzil Zaheer. 2024. A fresh take on stale embeddings: improving dense retriever training with corrector networks. *arXiv preprint arXiv:2409.01890*.

Nicholas Monath, Manzil Zaheer, Kelsey Allen, and Andrew McCallum. 2023. Improving dual-encoder training through dynamic indexes for negative mining. In *International Conference on Artificial Intelligence and Statistics*, pages 9308–9330. PMLR.

Mohammad Norouzi, Ali Punjani, and David J Fleet. 2013. Fast exact search in hamming space with multi-index hashing. *IEEE transactions on pattern analysis and machine intelligence*, 36(6):1107–1119.

Ankit Singh Rawat, Jiecao Chen, Felix Xinnan X Yu, Ananda Theertha Suresh, and Sanjiv Kumar. 2019. Sampled softmax with random fourier features. *Advances in Neural Information Processing Systems*, 32.

Sashank J Reddi, Satyen Kale, Felix Yu, Daniel Holtmann-Rice, Jiecao Chen, and Sanjiv Kumar. 2019. Stochastic negative mining for learning with large output spaces. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1940–1949. PMLR.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Devendra Singh Sachan, Mike Lewis, Mandar Joshi, Armen Aghajanyan, Wen-tau Yih, Joelle Pineau, and Luke Zettlemoyer. 2022. Improving passage retrieval with zero-shot question generation. *arXiv preprint arXiv:2204.07496*.

Nandan Thakur, Nils Reimers, and Jimmy Lin. 2022. Domain adaptation for memory-efficient dense retrieval. *arXiv preprint arXiv:2205.11498*.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *arXiv preprint arXiv:2007.00808*.

Shicheng Xu, Liang Pang, Jun Xu, Huawei Shen, and Xueqi Cheng. 2024. List-aware reranking-truncation joint model for search and retrieval-augmented generation. In *Proceedings of the ACM on Web Conference 2024*, pages 1330–1340.

Zhen Yang, Zhou Shao, Yuxiao Dong, and Jie Tang. 2024. Trisampler: A better negative sampling principle for dense retrieval. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 9269–9277.

Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2021. Optimizing dense retrieval model training with hard negatives. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1503–1512.

Wayne Xin Zhao, Jing Liu, Ruiyang Ren, and Ji-Rong Wen. 2024. Dense text retrieval based on pretrained language models: A survey. *ACM Transactions on Information Systems*, 42(4):1–60.