# In-Batch Negatives for Knowledge Distillation with Tightly-Coupled Teachers for Dense Retrieval

Sheng-Chieh Lin*, Jheng-Hong Yang* and Jimmy Lin

David R. Cheriton School of Computer Science
University of Waterloo

## Abstract

We present an efficient training approach to text retrieval with dense representations that applies knowledge distillation using the ColBERT late-interaction ranking model. Specifically, we propose to transfer the knowledge from a bi-encoder teacher to a student by distilling knowledge from ColBERT's expressive MaxSim operator into a simple dot product. The advantage of the bi-encoder teacher–student setup is that we can efficiently add in-batch negatives during knowledge distillation, enabling richer interactions between teacher and student models. In addition, using ColBERT as the teacher reduces training cost compared to a full cross-encoder. Experiments on the MS MARCO passage and document ranking tasks and data from the TREC 2019 Deep Learning Track demonstrate that our approach helps models learn robust representations for dense retrieval effectively and efficiently.

## 1 Introduction

For well over half a century, solutions to the *ad hoc* retrieval problem—where the system's task is return a list of top $k$ texts from an arbitrarily large corpus $\mathcal{D}$ that maximizes some metric of quality such as average precision or NDCG—has been dominated by *sparse* vector representations, for example, bag-of-words BM25. Even in modern multi-stage ranking architectures, which take advantage of large pretrained transformers such as BERT (Devlin et al., 2019), the models are deployed as *rerankers* over initial candidates retrieved based on sparse vector representations; this is sometimes called "first-stage retrieval". One well-known example of this design is the BERT-based reranker of Nogueira and Cho (2019); see Lin et al. (2020) for a recent survey.

The standard reranker architecture, while effective, exhibits high query latency, on the order of seconds per query (Hofstätter and Hanbury, 2019; Khattab and Zaharia, 2020) because expensive neural inference must be applied at query time on query–passage pairs. This design is known as a cross-encoder (Humeau et al., 2020), which exploits query–passage attention interactions across all transformer layers. As an alternative, a bi-encoder design provides an approach to ranking with dense representations that is far more efficient than cross-encoders (Lee et al., 2019; Reimers and Gurevych, 2019; Khattab and Zaharia, 2020; Karpukhin et al., 2020; Luan et al., 2021; Xiong et al., 2021; Qu et al., 2020; Hofstätter et al., 2021). Prior to retrieval, the vector representations can be precomputed for each of the texts in a corpus. When retrieving texts in response to a given query, computationally expensive transformer inference is replaced by much faster approximate nearest neighbor (ANN) search (Liu et al., 2004; Malkov and Yashunin, 2020).

Recently, researchers have proposed bi-encoders that produce multiple vectors to represent a query (or a passage) (Humeau et al., 2020; Luan et al., 2021; Khattab and Zaharia, 2020), which have proven to be effective both theoretically and empirically. However, the main disadvantage of these designs is their high storage requirements. For example, ColBERT (Khattab and Zaharia, 2020) requires storing all the WordPiece token vectors of each text (passage) in the corpus. On the MS MARCO passage corpus comprising 8.8M passages, for example, this requires 154 GiB.

Of course, a common alternative is to produce single vectors for queries and passages (Reimers and Gurevych, 2019). Although this design is less storage-demanding, it sacrifices ranking effectiveness since its structure breaks rich interactions between queries and passages compared to

*Contributed equally.



*Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*, pages 163–173
Bangkok, Thailand (Online), August 6, 2021. 

multi-vector bi-encoders or cross-encoders. Hence, improving the effectiveness of single-vector bi-encoders represents an important problem.

One approach to improving the effectiveness of single-vector bi-encoders is hard negative mining, by training with carefully selected negative examples that emphasize discrimination between relevant and non-relevant texts. There are several approaches to accomplish this. Karpukhin et al. (2020) and Qu et al. (2020) leverage large in-batch negatives to enrich training signals. Guu et al. (2020) and Xiong et al. (2021) propose to mine hard negatives using the trained bi-encoder itself. By searching for global negative samples from an asynchronously updated ANN index, the bi-encoder can learn information not present in the training data produced by sparse representations (Xiong et al., 2021). However, both large in-batch negative sampling and asynchronous ANN index updates are computationally demanding. The later is especially impractical for large corpora since it requires periodic inference over *all* texts in the corpus to ensure that the best negative examples are retrieved.

There is also work that explores knowledge distillation (KD) (Hinton et al., 2015) to enhance retrieval effectiveness and efficiency. Most related to our study is Hofstätter et al. (2020), who demonstrate that KD using a cross-encoder teacher significantly improves the effectiveness of bi-encoders for dense retrieval. Similarly, Barkan et al. (2020) investigate the effectiveness of distilling a trained cross-encoder into a bi-encoder for sentence similarity tasks. Gao et al. (2020a) explore KD combinations of different objectives such as language modeling and ranking. However, the above papers use computationally expensive cross-encoder teacher models; thus, combining them for KD with more advanced negative sampling techniques can be impractical.

In light of existing work on hard negative mining and knowledge distillation, we propose to improve the effectiveness of single-vector bi-encoders with a more efficient KD approach: in-batch KD using a bi-encoder teacher. The advantage of our design is that, during distillation, it enables the efficient exploitation of all possible query–passage pairs within a minibatch, which we call *tight coupling* (illustrated in Figure 1). This is a key difference between our KD approach and previous methods for dense retrieval, where only the scores of given query–passage triplets (not all combinations) are

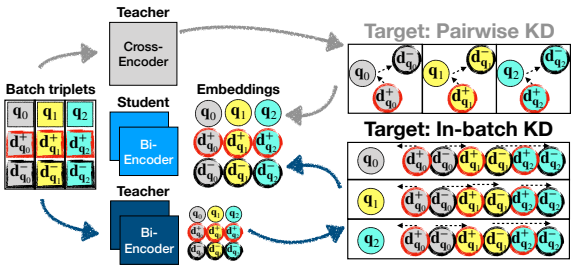



Figure 1: Illustration of the differences between pairwise knowledge distillation and our proposed in-batch knowledge distillation.

computed due to the computational costs of cross-encoders (Hofstätter et al., 2020; Gao et al., 2020a; Barkan et al., 2020).

The contribution of this work is a simple technique for efficiently adding in-batch negative samples during knowledge distillation when training a single-vector bi-encoder. For the remainder of this paper, we refer to this technique as "in-batch KD" for convenience. We empirically show that our model, even trained with BM25 negatives, can be more effective than cross-encoder teachers. With hard negatives, our method approaches the state of the art in dense retrieval. Our in-batch KD technique is able to incorporate hard negatives in a computationally efficient manner, without requiring large amounts of GPU memory for large batch sizes or expensive periodic index refreshes.

## 2 Background

We focus on improving the training efficiency and retrieval effectiveness of dense retrieval and begin by formalizing it as a dense representation learning problem. To be more specific, we propose to use knowledge distillation to enrich training signals and stabilize the representation learning procedure of bi-encoder models in the context of the well-known Noise-Contrastive Estimation (NCE) framework.

### 2.1 Dense Retrieval with Bi-encoders

The bi-encoder design has been widely adopted for dense retrieval (Lee et al., 2019; Chang et al., 2020; Guu et al., 2020; Karpukhin et al., 2020; Luan et al., 2021; Qu et al., 2020; Xiong et al., 2021), where queries and passages are encoded in a low-dimensional space. It aims to learn low-dimensional representations that pull queries and relevant passages together and push queries and non-relevant passages apart.

Following the work of Mnih and Kavukcuoglu

(2013), we formulate a common objective for dense representation learning for passage retrieval. Given a query $q$ and a parameterized scoring function $\phi_\theta$ that computes the relevance between a query and a candidate passage $p$, we define a probability distribution over documents in a corpus $\mathcal{D}$ with respect to relevance, as follows:

$$P^q_\theta(p, \mathcal{D}) = \frac{\exp(\phi_\theta(q,p))}{\sum_{p' \in \mathcal{D}} \exp(\phi_\theta(q,p'))} = \frac{\exp(\mathbf{h}_q \cdot \mathbf{h}_p)}{\sum_{p' \in \mathcal{D}} \exp(\mathbf{h}_q \cdot \mathbf{h}_{p'})}, \quad (1)$$

where $\mathbf{h}_q$ ($\mathbf{h}_p$) $\in \mathbb{R}^d$ denotes the query (passage) representation produced by the bi-encoder. A typical bi-encoder uses a simple scoring function for $\phi_\theta$, for example, the inner product of two vectors, as shown above.

The main challenge of evaluating and computing gradients of Eq. (1) is the prohibitively expensive computation cost given the number of passages in the corpus $\mathcal{D}$, typically millions (or even more). This is already setting aside the cost of using pretrained transformers such as BERT as the encoder to compute $\mathbf{h}_q$ and $\mathbf{h}_p$.

Thus, previous work approximates Eq. (1) by NCE, which samples $p \in \mathcal{D}^+$ from training data and $p' \in \mathcal{D}' = \{\mathcal{D}^+ \cup \mathcal{D}^-\}$, where $\mathcal{D}^-$ is from a noisy distribution such as candidates retrieved by BM25 (Nogueira and Cho, 2019), filtered by fine-tuned transformers (Qu et al., 2020), or retrieved by an asynchronously updated bi-encoder model itself (Xiong et al., 2021). Another simple yet effective approach is in-batch negative sampling, as used by Karpukhin et al. (2020), which takes $p$ and $p'$ of other queries within a minibatch as negative examples in NCE.

### 2.2 Knowledge Distillation

Other than designing sophisticated sampling methods for $p'$, training bi-encoder models using knowledge distillation (KD) with effective teacher models is another promising approach (Hofstätter et al., 2020). In this case, we aim to make the bi-encoder model mimic the teacher model's probability distribution as follows:

$$P^q_{\theta;\text{student}}(p, \mathcal{D}') = \frac{\exp(\mathbf{h}_q \cdot \mathbf{h}_p)}{\sum_{p' \in \mathcal{D}'} \exp(\mathbf{h}_q \cdot \mathbf{h}_{p'})} \approx \frac{\exp(\phi_{\hat{\theta}}(q,p)/\tau)}{\sum_{p' \in \mathcal{D}'} \exp(\phi_{\hat{\theta}}(q,p')/\tau)} = P^q_{\hat{\theta};\text{teacher}}(p, \mathcal{D}'), \quad (2)$$

where $\phi_{\hat{\theta}}$ denotes the relevance score estimated by a pretrained model parameterized by $\hat{\theta}$ and $\tau$, the temperature hyperparameter used in the KD framework. To improve retrieval effectiveness, one can leverage pre-computed scores from pretrained models such as cross-encoders, e.g., BERT, bi-encoders, e.g., ColBERT, or ensembled scores from multiple models $\phi_{\hat{\theta}} = \sum_j \phi_{\hat{\theta};j}$.

## 3 Our Approach

### 3.1 In-batch Knowledge Distillation

Using KD in Eq. (2) provides soft labels for bi-encoder training, and can be integrated with the previously mentioned NCE framework. In this work, we propose to enhance teacher–student interactions by adding in-batch negatives to our knowledge distillation. Specifically, we estimate $\phi_\theta$ on in-batch examples from a minibatch $\mathcal{B}$ guided by an auxiliary teacher model $\phi_{\hat{\theta}}$ through the minimization of Kullback–Leibler (KL) divergence of the two distributions:

$$\arg\min_\theta \sum_{q \in \mathcal{Q}_\mathcal{B}} \sum_{p \in \mathcal{D}'_\mathcal{B}} \mathcal{L}_{\phi_\theta, \phi_{\hat{\theta}}}, \quad (3)$$

where $\mathcal{L}_{\phi_\theta, \phi_{\hat{\theta}}}$ is:

$$P^q_{\hat{\theta};\text{teacher}}(p, \mathcal{D}'_\mathcal{B}) \log \frac{P^q_{\hat{\theta};\text{teacher}}(p, \mathcal{D}'_\mathcal{B})}{P^q_{\theta;\text{student}}(p, \mathcal{D}'_\mathcal{B})}. \quad (4)$$

Note that here we consider all pairwise relationship between queries and passages within a minibatch that contains a query set $\mathcal{Q}_\mathcal{B}$ and a passage set $\mathcal{D}'_\mathcal{B}$.

### 3.2 Teacher Model Choice

A **cross-encoder** has been shown to be an effective teacher (Hofstätter et al., 2020; Gao et al., 2020a) since it allows rich interactions between the intermediate transformer representations of a query $q$ and a passage $p$. For example, a "vanilla" cross-encoder design using BERT can be denoted as:

$$\phi_{\hat{\theta};\text{Cat}} \triangleq W f(\mathbf{h}_{q \oplus p}), \quad (5)$$

where the ranking score is first computed by the hidden representation of the concatenation $q \oplus p$ from BERT (along with the standard special tokens) and then mapped to a scalar by a pooling operation $f$ and a mapping matrix $W$.

Although effective, due to BERT's quadratic complexity with respect to input sequence length, this design makes exhaustive combinations between a query and possible candidates impractical,

since this requires evaluating cross-encoders $|\mathcal{B}|^2$ times to compute Eq. (3) using Eq. (5). Thus, an alternative is to conduct *pairwise* KD by computing the KL divergence of only two probabilities of a positive pair $(q, p)$ and a negative pair $(q, p')$ for each query $q$. However, this might not yield a good approximation of Eq. (2).

A **bi-encoder** can also be leveraged as a teacher model, which has the advantage that it is more feasible to perform exhaustive comparisons between queries and passages since they are passed through the encoder independently. Among bi-encoder designs, ColBERT is a representative model that uses late interactions of multiple vectors $(\{\mathbf{h}_q^1, \ldots, \mathbf{h}_q^i\}, \{\mathbf{h}_p^1, \ldots, \mathbf{h}_p^j\})$ to improve the robustness of dense retrieval, as compared to inner products of pairs of single vectors $(\mathbf{h}_q, \mathbf{h}_p)$. Specifically, Khattab and Zaharia (2020) propose the following fine-grained scoring function:

$$\phi_{\hat{\theta};\text{MaxSim}} \triangleq \sum_{i \in |\mathbf{h}_q|} \max_{j \in |\mathbf{h}_p|} \mathbf{h}_q^i \cdot \mathbf{h}_p^j, \qquad (6)$$

where $i$ and $j$ are the indices of token representations of a query $q$ and a passage $p$ of ColBERT (Khattab and Zaharia, 2020).

The contribution of our work is in-batch knowledge distillation with a tightly-coupled teacher. The computation of $\phi_{\hat{\theta};\text{MaxSim}}$ enables exhaustive inference over all query–passage combinations in the minibatch $\mathcal{B}$ with only $2{\cdot}|\mathcal{B}|$ computation cost, enabling enriched interactions between teacher and student. We call this design **T**ightly-**C**oupled **T**eacher ColBERT (TCT-ColBERT). Table 1 provides a training cost comparison between different teachers. When training with pairwise KD, cross-encoders exhibit the highest training cost. On the other hand, ColBERT enables in-batch KD at a modest training cost compared to pairwise KD.

TCT-ColBERT provides a flexible design for bi-encoders, as long as the encoders produce query and passage representations independently. For simplicity, our student model adopts shared encoder weights for both the query and the passage, just like the teacher model ColBERT. Following Khattab and Zaharia (2020), for each query (passage), we prepend the `[CLS]` token and another special `[Q]` (`[D]`) token in the input sequence for both our teacher and student models. The student encoder outputs single-vector dense representations $(\mathbf{h}_q, \mathbf{h}_p)$ by performing average pooling over the token embeddings from the final layer.

Table 1: Training cost comparison. We report the training time per batch against the baseline (without a teacher model) on a single TPU-v2. Our backbone model is BERT-base, with batch size 96. The in-batch cross-encoder training time is not available because it exceeds the memory limit.

| Teacher / KD strategy | Pairwise | In-batch |
|---|---|---|
| Cross-encoder ($\phi_{\hat{\theta};\text{Cat}}$) | +48.1% | OOM |
| ColBERT ($\phi_{\hat{\theta};\text{MaxSim}}$) | +32.7% | +33.5% |

### 3.3 Hard Negative Sampling

Given that in-batch negative sampling is an efficient way to add more information into knowledge distillation, we wonder whether our tightly-coupled teacher design works well when applied to more sophisticated sampling methods. Following the work of Xiong et al. (2021), we use our pretrained bi-encoder model, namely TCT-ColBERT, to encode the corpus and sample "hard" negatives for each query to create new training triplets by using the negatives $\mathcal{D}^-$ of the bi-encoder instead of BM25. Specifically, we explore three different training strategies:

1. **HN**: we train the bi-encoder using in-batch hard negatives without the guide of ColBERT.
2. **TCT HN**: we train the bi-encoder with TCT-ColBERT;
3. **TCT HN+**: we first fine-tune our ColBERT teacher with augmented training data containing hard negatives and then distill its knowledge into the bi-encoder student through TCT-ColBERT.

We empirically explore the effectiveness of these strategies for both passage and document retrieval.

## 4 Experiments

In this section, we conduct experiments on the MS MARCO passage and document corpora. For passage ranking, we first train models on BM25 negatives as warm-up and compare different KD methods. We then further train models on the hard negatives retrieved by the BM25 warmed-up checkpoint. For document ranking, following previous work (Xiong et al., 2021; Zhan et al., 2020; Lu et al., 2021), we start with our BM25 warmed-up checkpoint for passage ranking and conduct additional hard negative training.

Table 2: Passage retrieval results with BM25 negative training. For knowledge distillation (KD) methods, the effectiveness of teacher (T) models is also reported. All our implemented models are labeled with a number and superscripts represent significant improvements over the labeled model (paired $t$-test, $p < 0.05$).

| Strategy | Model | # params of Teacher | MARCO Dev | | TREC-DL '19 | |
|---|---|---|---|---|---|---|
| | | | MRR@10 (T/S) | R@1K | NDCG@10 (T/S) | R@1K |
| - | (1) Baseline | - | - / .310 | .945 | - / .626 | .658 |
| Pairwise KD | KD-T1 (Hofstätter et al., 2020) | 110M | .376 / .304 | .931 | .730 / .631 | .702 |
| | KD-T2 (Hofstätter et al., 2020) | 467M | .399 / .315 | .947 | .743 / .668 | .737 |
| | (2) KD-T2 (Ours) | 467M | .399 / .341[1] | .964[1] | .743 / .659[1] | .708[1] |
| | (3) KD-ColBERT | 110M | .350 / .339[1] | .962[1] | .730 / .670[1] | .710[1] |
| In-batch KD | (4) TCT-ColBERT | 110M | .350 / **.344**[1,3] | **.967**[1,3] | .730 / **.685**[1] | **.745**[1,2,3] |

### 4.1 Passage Retrieval

We perform *ad hoc* passage retrieval on the MS MARCO passage ranking dataset (Bajaj et al., 2016), which consists of a collection of 8.8M passages from web pages and a set of ∼0.5M relevant (query, passage) pairs as training data. We evaluate model effectiveness on two test sets of queries:

1. MARCO Dev: the development set of MS MARCO comprises 6980 queries, with an average of one relevant passage per query.
2. TREC-DL '19 (Craswell et al., 2019): the organizers of the Deep Learning Track at the 2019 Text REtrieval Conference (TREC) released 43 queries with multi-graded (0–3) relevance labels on 9K (query, passage) pairs.

To evaluate output quality, we report MRR@10 (NDCG@10) for MARCO Dev (TREC-DL '19) and Recall@1K, denoted as R@1K. To compare with current state-of-the-art models, we evaluate our design, TCT-ColBERT, under two approaches for negative sampling: (1) BM25 and (2) hard negatives retrieved by the bi-encoder itself.

#### 4.1.1 Training with BM25 Negatives

In this setting, models are trained using the official public data `triples.train.small`, where negative samples are produced by BM25. We compare different bi-encoder models using BERT-base as the backbone, which uses single 768-dim vectors to represent each query and passage:

1. **Baseline**: a single-vector bi-encoder trained with in-batch negatives, as discussed in Section 2.1, which is similar to Karpukhin et al. (2020) but with a smaller batch size.
2. **Pairwise KD**: the approach of Hofstätter et al. (2020), who improve ranking effectiveness using cross-encoders with pairwise KD.

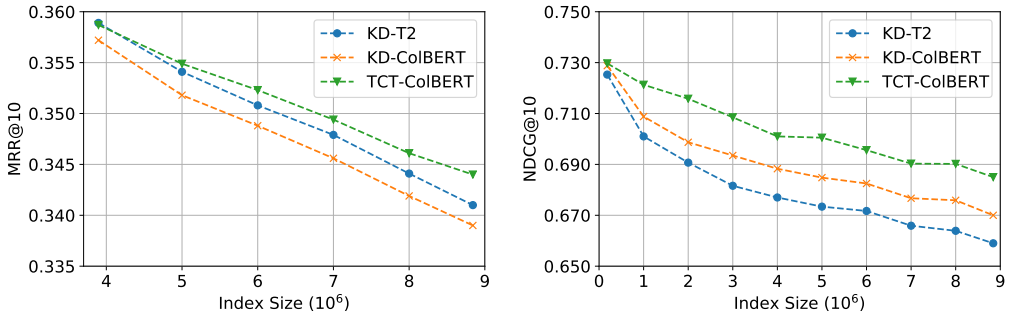
We also compare against two models, KD-T1 and KD-T2, which use BERT-base bi-encoders as student models. In the former, the student is distilled from a BERT-base cross-encoder, while the latter is distilled from ensembled cross-encoders comprising BERT-base, BERT-large, and ALBERT-large. These figures reported in Table 2 are copied from Hofstätter et al. (2020). For a fair comparison with our models based on KL-divergence KD, we also implement our KD-T2 using the precomputed pairwise softmax probabilities provided by Hofstätter et al. (2020) (who use MSE margin loss for KD). In addition, we adopt *pairwise* softmax probabilities from fine-tuned ColBERT to train KD-ColBERT for comparison.

All our models are fine-tuned with batch size 96 and learning rate $7 \times 10^{-6}$ for 500K steps on a single TPU-V2. For TCT-ColBERT, there are two steps in our training procedure: (1) fine-tune $\phi_{\hat{\theta};\text{MaxSim}}$ as our teacher model, (2) freeze $\phi_{\hat{\theta};\text{MaxSim}}$ and distill knowledge into our student model $\phi_\theta$. We keep all the hyperparameter settings the same but adjust temperature $\tau = 0.25$ for KD at the second step. For all our models, including the baseline, we initialize the student model using the fine-tuned weights of the teacher model in the first step. We limit the input tokens to 32 (150) for queries (passages). To evaluate effectiveness, we encode all passages in the corpus and conduct brute force search over the vector representations.

Our main results, including paired $t$-test for significance testing, are shown in Table 2. In addition to the effectiveness of the student models, we also show the effectiveness of the teacher models for the KD methods.[1]

First, we see that pairwise KD methods show significant improvements over the baseline, indicat-

[1]We report our trained ColBERT's accuracy by reranking the top-1000 candidates provided officially.




(a) MARCO Dev

(b) TREC-DL '19

Figure 2: Passage retrieval effectiveness on a synthetic corpus comprising relevant passages and BM25 results as additional "distractors" randomly sampled from the corpus are added.

ing that information from BM25 negatives cannot be fully exploited without teacher models. Second, although KD-T2 improves the bi-encoder's effectiveness over KD-T1, it is not consistently better than KD-ColBERT in terms of students' effectiveness. We suspect that they have comparable capabilities to discriminate most paired passages (BM25 negative vs. positive samples), i.e., ColBERT is good enough to guide bi-encoder student models to discriminate them. On the other hand, our TCT-ColBERT model, which uses only one teacher model and adds only 33% more training time over the baseline, yields the best effectiveness, demonstrating the advantages of our proposed in-batch KD — exhaustive exploitation of all query–document combinations in a minibatch.

To understand why TCT-ColBERT yields better results, we study the models' retrieval effectiveness against carefully selected distractors. We start with a small synthetic corpus composed of the relevant passages and the top-1000 BM25 candidates of the 6980 (43) queries from MARCO Dev (TREC-DL '19). To increase the corpus size, we gradually add passages uniformly sampled from the corpus without replacement. From Figure 2, we see that the three KD models exhibit nearly the same effectiveness when the corpus only contains BM25 candidates. This shows that the bi-encoders learn to discriminate relevant passages from the BM25 negative samples well. However, as the index size increases, TCT-ColBERT demonstrates better ranking effectiveness than the other pairwise KD methods, indicating that the learned representations are more robust. We attribute this robustness against "distractors" to the enriched information from in-batch KD, where we are able to exploit all in-batch query–document combinations.

#### 4.1.2 Training with Hard Negatives

In this subsection, we evaluate TCT-ColBERT when training with hard negatives (HNs). We compare our model to four competitive approaches:

1. **ANCE** (Xiong et al., 2021) is the most representative work, which proposes asynchronous index refreshes to mine hard negatives. The model is trained for 600K steps with index refreshes every 10K steps. ANCE uses RoBERTa-base as its backbone.
2. **LTRe** (Zhan et al., 2020) further improves from an ANCE checkpoint by adding more training steps with the same hard negative mining approach; thus, the computation cost of index refreshes from ANCE cannot be neglected. LTRe also use RoBERTa-base as its backbone.
3. **SEED-Encoder** (Lu et al., 2021) leverages a pretraining strategy to enhance the capability of the bi-encoder, which is further fine-tuned with HNs using asynchronous index refreshes.
4. **RocketQA** (Qu et al., 2020) trains a bi-encoder model using hard negatives denoised by a cross-encoder, ERNIE-2.0-Large (Sun et al., 2019). It further demonstrates that training bi-encoders with many in-batch negatives (batch size up to 4096) significantly improves ranking effectiveness; however, this approach is computationally expensive (the authors report using 8×V100 GPUs for training). To the best of our knowledge, RocketQA represents the state of the art in single-vector bi-encoders for dense retrieval. For a more fair comparison, we also report the ranking effectiveness of their model trained with a smaller batch size of 128.

For all the approaches above, we directly copy the reported effectiveness from the original papers.

Table 3: Passage retrieval results with hard negative training. All our implemented models are labeled with a number and superscripts represent significant improvements over the labeled model (paired $t$-test, $p < 0.05$).

| Model | # Index Refresh | Batch Size | MARCO Dev | | TREC-DL '19 | |
|---|---|---|---|---|---|---|
| | | | MRR@10 | R@1K | NDCG@10 | R@1K |
| ANCE (Xiong et al., 2021) | 60 | 32 | .330 | .959 | .648 | - |
| LTRe (Zhan et al., 2020) | 60 | 32 | .341 | .962 | .675 | - |
| SEED-Encoder (Lu et al., 2021) | $\geq$10 (est.) | - | .339 | .961 | - | - |
| RocketQA (Qu et al., 2020) | 1 | 128 | .310 | - | - | - |
| RocketQA (Qu et al., 2020) | 1 | 4096 | **.364** | - | - | - |
| (1) TCT-ColBERT | 0 | 96 | .344 | .967 | .685 | .745 |
| (2) w/ HN | 1 | 96 | .237 | .929 | .543 | .674 |
| (3) w/ TCT HN | 1 | 96 | .354$^{1,2}$ | **.971**$^{1,2}$ | .705$^{2}$ | **.765**$^{1,2}$ |
| (4) w/ TCT HN+ | 1 | 96 | .359$^{1,2}$ | .970$^{1}$ | **.719**$^{1,2}$ | .760$^{1}$ |

For our TCT-ColBERT model, following the settings of the above approaches, we first use our TCT-ColBERT model trained on BM25 negatives as a warm-up starting point and index all 8.8M MARCO passages. Using the warmed-up index, we retrieve top-200 passages for each training query and randomly sample (with replacement) hard negatives from the 200 candidates to form our training data. Note that due to resource limitations we do not conduct experiments with asynchronous index refreshes since multiple V100 GPUs are required for such a model training scheme.[2] In this experiment, all the hyperparameter settings are the same as the ones in the BM25 negative training, except for training steps, which is set to 100K for both student and teacher training.

Table 3 reports the results of our experiments with hard negative training. First, we observe that our TCT-ColBERT model trained with BM25 negatives marginally outperforms the other models trained with HNs, except for RocketQA. Comparing the different training strategies discussed in Section 3.3 (second main block of the table), we see that the ranking effectiveness of TCT-ColBERT (HN) degrades when training on hard negatives without the guide of a teacher. This is consistent with the findings of Qu et al. (2020) that hard negatives contain noisy information (i.e., some hard negatives may actually be relevant). Also, Xiong et al. (2021) show that training bi-encoders with hard negatives can be unstable: hard negatives benefit ranking effectiveness only under certain hyperparameter settings.

In contrast, hard negative training using ColBERT's in-batch KD further boosts ranking effectiveness, especially when our teacher (ColBERT) is trained with the same hard negative samples beforehand. It is also worth noting that our TCT-ColBERT (w/ TCT HN+) with batch size 96 yields competitive ranking effectiveness compared to RocketQA (the current state of the art), which uses batch size 4096. These results demonstrate the advantages of our TCT design: our approach effectively exploits hard negatives in a computationally efficient manner (i.e., without the need for large batch sizes or periodic index refreshes).

### 4.2 Document Retrieval

To validate the effectiveness and generality of our training strategy, we conduct further experiments on document retrieval using the MS MARCO document ranking dataset. This dataset contains 3.2M web pages gathered from passages in the MS MARCO passage ranking dataset. Similar to the passage condition, we evaluate model effectiveness on two test sets of queries:

1. MARCO Dev: the development set contains 5193 queries, each with exactly one relevant document.
2. TREC-DL '19: graded relevance judgments are available from the TREC 2019 Deep Learning Track, but on only 43 queries.

Per official guidelines, we report different metrics for the two query sets: MRR@100 for MARCO Dev and NDCG@10 for TREC-DL '19.

Following the FirstP setting for document retrieval described in Xiong et al. (2021), we feed the first 512 tokens of each document for encoding, and start with the warmed-up checkpoint for our encoder's parameters trained for passage retrieval (using BM25 negatives, as described in Section 4.1.1). The settings for fine-tuning our warmed-up encoder

[2]Re-encoding the entire corpus takes $\sim$10 hours on one GPU.

Table 4: Document retrieval results using the FirstP approach. All our implemented models are labeled with a number and superscripts represent significant improvements over the labeled model (paired $t$-test, $p < 0.05$).

| Model | MARCO Dev | TREC-DL '19 |
|---|---|---|
| | MRR@100 | NDCG@10 |
| ANCE (Xiong et al., 2021) | .368 | .614 |
| LTRe (Zhan et al., 2020) | - | .634 |
| SEED-Encoder (Lu et al., 2021) | .394 | - |
| (1) TCT-ColBERT | .339 | .573 |
| (2) w/ TCT HN+ | .392[1] | .613 |
| (3) w/ 2× TCT HN+ | **.418**[1,2] | **.650**[1,2] |

(e.g., learning rate, training steps, top-200 negative sampling) are the same as passage retrieval except for batch size, which is set to 64.

Ranking effectiveness is reported in Table 4. First, we observe that TCT-ColBERT (our warmed-up checkpoint) performs far worse than other approaches to document retrieval using the FirstP method. This may be due to the fact that FirstP document retrieval is very different from passage retrieval, making zero-shot transfer ineffective. After applying HN training on both teacher and student models (condition 2), the ranking effectiveness increases significantly. In addition, we find that another iteration of training with an index refresh (condition 3) further improves ranking effectiveness. To sum up, in the document ranking task, TCT-ColBERT yields competitive effectiveness with a one-time index refresh and outperforms other computationally expensive methods with one additional index refresh.

### 4.3 Dense–Sparse Hybrids

In our final set of experiments, we show that dense retrieval with single-vector representations can be integrated with results from sparse retrieval to further increase effectiveness. We illustrate the end-to-end tradeoffs in terms of quality, time, and space of different dense–sparse hybrid combinations on the passage retrieval tasks.

Many papers (Luan et al., 2021; Gao et al., 2020b; Ma et al., 2021; Lin et al., 2021) have demonstrated that sparse retrieval can complement dense retrieval via a simple linear combination of their scores. In our implementation, for each query $q$, we use sparse and dense techniques to retrieve the top-1000 passages, $\mathcal{D}_{sp}$ and $\mathcal{D}_{ds}$, with their relevance scores, $\phi_{sp}(q, p \in \mathcal{D}_{sp})$ and $\phi_{ds}(q, p \in \mathcal{D}_{ds})$, respectively. Then, we compute the final relevance score for each retrieved passage $\phi(q, p)$, where $p \in \mathcal{D}_{sp} \cup \mathcal{D}_{ds}$, as follows:

$$\begin{cases} \alpha \cdot \phi_{sp}(q,p) + \min\limits_{p \in \mathcal{D}_{ds}} \phi_{ds}(q,p), & \text{if } p \notin D_{ds} \\ \alpha \cdot \min\limits_{p \in \mathcal{D}_{sp}} \phi_{sp}(q,p) + \phi_{ds}(q,p), & \text{if } p \notin D_{sp} \\ \alpha \cdot \phi_{sp}(q,p) + \phi_{ds}(q,p), & \text{otherwise.} \end{cases}$$

This technique is an approximation of a linear combination of sparse and dense retrieval scores. Specifically, if $p \notin \mathcal{D}_{sp}$(or $\mathcal{D}_{ds}$), we instead use the minimum score of $\phi_{sp}(q, p \in \mathcal{D}_{sp})$, or $\phi_{ds}(q, p \in \mathcal{D}_{ds})$ as a substitute.

For the sparse and dense retrieval combinations, we tune the hyperparameter $\alpha$ on 6000 randomly sampled queries from the MS MARCO training set. We conduct dense–sparse hybrid experiments with sparse retrieval (BM25 ranking) on the original passages (denoted BM25) and on passages with docTTTTTquery document expansion (Nogueira and Lin, 2019) (denoted doc2query-T5). To characterize end-to-end effectiveness and efficiency, we perform sparse retrieval with the Pyserini toolkit (Lin et al., 2021) and dense retrieval with Faiss (Johnson et al., 2017), but implement the score combination in separate custom code.

Table 5 shows passage retrieval results in terms of ranking effectiveness, query latency, and storage requirements (i.e., index size) for each model and Table 6 reports the component latencies of our TCT-ColBERT dense–sparse hybrid.[3] The cross-encoder reranker of Nogueira and Cho (2019) provides a point of reference for multi-stage reranking designs, which is effective but slow.

Generally, dense retrieval methods (whether single-vector or multi-vector) are more effective but slower than sparse retrieval methods, which rely on bag-of-words querying using inverted indexes. Single-vector dense models also require more space than sparse retrieval methods. Moving

[3]Here we assume running dense and sparse retrieval in parallel.

Table 5: End-to-end comparisons of output quality, query latency, and storage requirements for passage retrieval.

| | Ranking effectiveness | | Latency | Storage |
|---|---|---|---|---|
| | MARCO Dev | TREC-DL '19 | ms/q | GiB |
| **Sparse retrieval** | | | | |
| BM25 with Anserini (Yang et al., 2018) | .184 | .506 | 55 | 4 |
| DeepCT (Dai and Callan, 2020) | .243 | .551 | 55 | 4 |
| doc2query-T5 (Nogueira and Lin, 2019) | .277 | .551 | 64 | 14 |
| **Dense retrieval: single-vector** | | | | |
| TAS-B (Hofstätter et al., 2021) | .343 | .722 | 64 | 13 |
| RocketQA (Qu et al., 2020) | .370 | - | 107[b] | 13[a] |
| TCT-ColBERT | .344 | .685 | 107 | 13 |
| TCT-ColBERT (w/ TCT HN+) | .359 | .719 | 107 | 13 |
| **Dense retrieval: multi-vector** | | | | |
| ME-BERT (Luan et al., 2021) | .334 | .687 | - | 96 |
| ColBERT (Khattab and Zaharia, 2020) | .360 | - | 458 | 154 |
| **Hybrid dense + sparse** | | | | |
| CLEAR (Gao et al., 2020b) | .338 | .699 | - | 17[a] |
| ME-HYBRID-E (Luan et al., 2021) | .343 | .706 | - | 100 |
| TAS-B + doc2query-T5 (Hofstätter et al., 2021) | .360 | .753 | 67 | 27[a] |
| TCT-ColBERT + BM25 | .356 | .720 | 110 | 17 |
| TCT-ColBERT + doc2query-T5 | .366 | .734 | 110 | 27 |
| TCT-ColBERT (w/ TCT HN+) + BM25 | .369 | .730 | 110 | 17 |
| TCT-ColBERT (w/ TCT HN+) + doc2query-T5 | .375 | .741 | 110 | 27 |
| **Multi-stage reranking** | | | | |
| BM25 + BERT-large (Nogueira and Cho, 2019) | .365 | .736 | 3500 | 4 |
| TAS-B + doc2query-T5 + Mono-Duo-T5 (Hofstätter et al., 2021) | .421 | .759 | 12800 | 27[a] |
| RocketQA with reranking (Qu et al., 2020) | .439 | - | - | 13[a] |

[a] We estimate dense index size using 16-bit floats; for hybrid, we add the sizes of sparse and dense indexes.
[b] We assume latency comparable to our settings.

Table 6: Component latencies per query of our model.

| Stage | latency (ms) | device |
|---|---|---|
| BERT query encoder | 7 | GPU |
| Dot product search | 100 | GPU |
| Score combination | 3 | CPU |

from single-vector to multi-vector dense models, we see that ColBERT exhibits higher effectiveness but is slower and requires much more storage.

Finally, when integrated with sparse retrieval methods, TCT-ColBERT is able to beat a basic multi-stage reranking design (BM25 + BERT-large), but with much lower query latency, although at the cost of increased storage. Hybrid TCT-ColBERT (w/ TCT HN+) + doc2query-T5 compares favorably with a recent advanced model, TAS-B + doc2query-T5 (Hofstätter et al., 2021), which introduces topic-aware sampling and dual teachers, incorporating part of our TCT-ColBERT work. Nevertheless, even the best hybrid variant of TCT-ColBERT alone, without further reranking, remains quite some distance from RocketQA, the current state of the art (with reranking using cross-encoders). This suggests that there remain relevance signals that require full attention interactions to exploit.

## 5 Conclusions

Improving the effectiveness of single-vector bi-encoders is an important research direction in dense retrieval because of lower latency and storage requirements compared to multi-vector approaches. We propose a teacher–student knowledge distillation approach using tightly coupled bi-encoders that enables exhaustive use of query–passage combinations in each minibatch. More importantly, a bi-encoder teacher requires less computation than a cross-encoder teacher. Finally, our approach leads to robust learned representations.

Overall, our hard negative sampling strategy leads to an effective *and* efficient dense retrieval technique, which can be further combined with sparse retrieval techniques in dense–sparse hybrids. Together, these designs provide a promising solution for end-to-end text retrieval that balances quality, query latency, and storage requirements.

## Acknowledgements

This research was supported in part by the Canada First Research Excellence Fund and the Natural Sciences and Engineering Research Council (NSERC) of Canada.

## References


Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. 2016. MS MARCO: A human generated machine reading comprehension dataset. *arXiv:1611.09268*.

Oren Barkan, Noam Razin, Itzik Malkiel, Ori Katz, Avi Caciularu, and Noam Koenigstein. 2020. Scalable attentive sentence-pair modeling via distilled sentence embedding. In *Proc. AAAI*.

Wei-Cheng Chang, Felix X. Yu, Yin-Wen Chang, Yiming Yang, and Sanjiv Kumar. 2020. Pre-training tasks for embedding-based large-scale retrieval. In *Proc. ICLR*.

Nick Craswell, Bhaskar Mitra, and Daniel Campos. 2019. Overview of the TREC 2019 deep learning track. In *Proc. TREC*.

Zhuyun Dai and Jamie Callan. 2020. Context-aware term weighting for first stage passage retrieval. In *Proc. SIGIR*, page 1533–1536.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. NAACL*, pages 4171–4186.

Luyu Gao, Zhuyun Dai, and Jamie Callan. 2020a. Understanding BERT rankers under distillation. In *Proc. ICTIR*, pages 149–152.

Luyu Gao, Zhuyun Dai, Zhen Fan, and Jamie Callan. 2020b. Complementing lexical retrieval with semantic residual embedding. *arXiv:2004.13969*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. REALM: Retrieval-augmented language model pre-training. *arXiv:2002.08909*.

Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. In *Proc. NeurIPS: Deep Learning and Representation Learning Workshop*.

Sebastian Hofstätter, Sophia Althammer, Michael Schröder, Mete Sertkan, and Allan Hanbury. 2020. Improving efficient neural ranking models with cross-architecture knowledge distillation. *arXiv:2010.02666v2*.

Sebastian Hofstätter and Allan Hanbury. 2019. Let's measure run time! Extending the IR replicability infrastructure to include performance aspects. In *Proc. OSIRRC: CEUR Workshop*, pages 12–16.

Sebastian Hofstätter, Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy Lin, and Allan Hanbury. 2021. Efficiently teaching an effective dense retriever with balanced topic aware sampling. In *Proc. SIGIR*.

Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. 2020. Poly-encoders: Architectures and pre-training strategies for fast and accurate multi-sentence scoring. In *Proc. ICLR*.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with GPUs. *arXiv:1702.08734*.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proc. EMNLP*, pages 6769–6781.

Omar Khattab and Matei Zaharia. 2020. ColBERT: Efficient and effective passage search via contextualized late interaction over BERT. In *Proc. SIGIR*, page 39–48.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Proc. ACL*, pages 6086–6096.

Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021. Pyserini: A Python toolkit for reproducible information retrieval research with sparse and dense representations. In *Proc. SIGIR*.

Jimmy Lin, Rodrigo Nogueira, and Andrew Yates. 2020. Pretrained transformers for text ranking: BERT and beyond. *arXiv:2010.06467*.

Ting Liu, Andrew W. Moore, Alexander Gray, and Ke Yang. 2004. An investigation of practical approximate nearest neighbor algorithms. In *Proc. NeurIPS*, page 825–832.

Shuqi Lu, Chenyan Xiong, Di He, Guolin Ke, Waleed Malik, Zhicheng Dou, Paul Bennett, Tieyan Liu, and Arnold Overwijk. 2021. Less is more: Pretraining a strong siamese encoder using a weak decoder. *arXiv:2102.09206*.

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.

Xueguang Ma, Kai Sun, Ronak Pradeep, and Jimmy Lin. 2021. A replication study of dense passage retriever. *arXiv:2104.05740*.

Yu A. Malkov and D. A. Yashunin. 2020. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *Transactions on Pattern Analysis and Machine Intelligence*, 42(4):824–836.

Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In *Proc. NIPS*, pages 2265–2273.

Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with BERT. *arXiv:1901.04085*.

Rodrigo Nogueira and Jimmy Lin. 2019. From doc2query to docTTTTTquery.

Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2020. RocketQA: An optimized training approach to dense passage retrieval for open-domain question answering. *arxiv:2010.08191v1*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proc. EMNLP*, pages 3982–3992.

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2019. ERNIE 2.0: A continual pre-training framework for language understanding. *arXiv:1907.12412*.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2021. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *Proc. ICLR*.

Peilin Yang, Hui Fang, and Jimmy Lin. 2018. Anserini: Reproducible ranking baselines using Lucene. *Journal of Data and Information Quality*, 10(4):Article 16.

Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. 2020. Learning to retrieve: How to train a dense retrieval model effectively and efficiently. *arXiv:2010.10469*.