Fantastic (small) Retrievers and How to Train Them: MXBAI-EDGE-COLBERT-V0 Tech Report

Rikiya Takehi[1,2*], Benjamin Clavié[1], Sean Lee[1], and Aamir Shakir[1]

[1] Mixedbread AI
[2] Waseda University
{rikiya,ben,sean}@mixedbread.com

Abstract. In this work, we introduce mxbai-edge-colbert-v0 models, at two different parameter counts: 17M and 32M. As part of our research, we conduct numerous experiments to improve retrieval and late-interaction models, which we intend to distill into smaller models as proof-of-concepts. Our ultimate aim is to support retrieval at all scales, from large-scale retrieval which lives in the cloud to models that can run locally, on any device. mxbai-edge-colbert-v0 is a model that we hope will serve as a solid foundation backbone for all future experiments, representing the first version of a long series of small proof-of-concepts. As part of the development of mxbai-edge-colbert-v0, we conducted multiple ablation studies, of which we report the results. In terms of downstream performance, mxbai-edge-colbert-v0 is a particularly capable small model, outperforming ColBERTv2 on common short-text benchmarks (BEIR) and representing a large step forward in long-context tasks, with unprecedented efficiency.

1 Introduction

In the last two years, neural Information Retrieval (IR) has experienced an unprecedented level of interest, owing in large part to the rapid development and deployment of Large Language Models (LLMs) and the proven effectiveness of Retrieval Augmented Generation (RAG) pipelines [13], where retrieval models are used to provide LLMs with useful context.

As part of this wave, end-user interest in multi-vector retrieval methods, also called late interaction models or, more simply, ColBERT, after the model which initially introduced this method [11]. Where the dominant paradigm in neural IR, Dense Passage Retrieval (DPR) [36], leverages a single, large vector to represent documents, ColBERT models instead employ numerous smaller vectors, with each individual token representation projected to a small dimension then retained. In order to make this tractable, ColBERT models are frequently used with aggressive index quantization [25,24] or as second-stage rankers in a larger pipeline.

The growing popularity of multi-vector models can be explained by their retrieval performance. ColBERT models have been noted for their particularly

* Work performed during an internship at Mixedbread.

robust out-of-domain performance [25], especially in multi-modal settings [27]. They have also recently been demonstrated to provably alleviate certain limitations of single-vector retrieval approaches, with a 150M parameter ColBERT models vastly outperforming 8B parameter single-vector embeddings on benchmarks designed to test the limits of embedding models [34].

In spite of these strong performances, the ecosystem for open ColBERT models have moved more slowly than that of single-vector models. Up until last year, the most widely used ColBERT model was ColBERTv2, originally released in 2021. Subsequently, answerai-colbert-small-v1[1] demonstrated a 33 million parameter ColBERT model could outperform all existing small retrievers, and reached performance exceeding even that of ColBERTv2 and most <500M parameter retrievers.

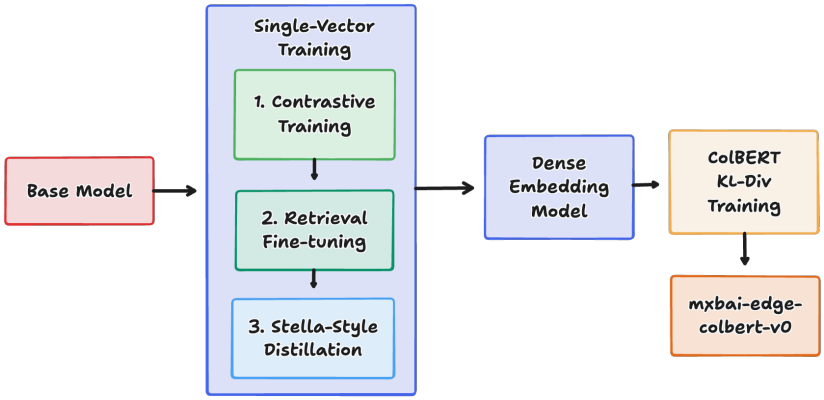

Fig. 1. An overview of the full training process

However, there has, until very recently, been a lack of late-interaction models featuring modern features, such as long context handling due to backbone limitations, at least in the text modality. Indeed, both ColBERTv2 and answerai-colbert were built on top of BERT variants, namely the original BERT [6] and MiniLM [32], with short context limits and poor efficiency, especially across longer contexts.

ModernBERT [33] spearheaded a new wave of novel encoders, built with efficiency in mind and allowing long-context encoders. Following its original release, it has been followed by Ettin, a reproduction of it across model sizes, and ModernVBERT, which combines Ettin with a vision-encoder, bringing the architectural improvements to multi-modality. GTE-ModernColBERT[2] [1] was subsequently released, leveraging ModernBERT as a backbone, and creating the new

[1] `https://huggingface.co/answerdotai/answerai-colbert-small-v1`
[2] `lightonai/GTE-ModernColBERT-v1`

de-facto standard for 130M+ parameter ColBERT, outperforming ColBERTv2 and all dense retrievers in its parameter class.

A large gap, however, remains: while GTE-ModernColBERT is a strong, "full-sized" model, answerai-colbert-small-v1 remains, by far, the most downloaded ColBERT model, in spite of its architectural limitations. In our own exploratory work at Mixedbread, we found ourselves frequently using it, as its small size and strong performance provided a strong testbed for various experiments. We firmly believe that performance at both ends of the scale spectrum is very important, especially as small models are strong predictor of the performance impact of model modifications.

As such, we decided to address this gap, and create the mxbai-edge-colbert-v0 family of ColBERT models. These models come in two different sizes, with 17 and 32 million parameters, and have been created to serve as a strong baseline to support further experiments while addressing the needs of users seeking a modern, low parameter-count ColBERT. To train these models, we first created dense embedding baselines through a series of three training stages, before running numerous ablations resulting in the released models.

The resulting models are strong performers across the board, with considerably improved efficiency over previous models. Notably, mxbai-edge-colbert-v0-17m outperforms ColBERTv2 despite an embedding dimension of 64, half of the commonly used 128, and an extremely low compute and memory footprint. Its strong performance, combined with long-context handling and very low latencies, make it particularly suitable for re-ranking applications on-device.

These models, the first of an hopefully long series of efficient edge models, represent a solid foundation for further studies onto the effectiveness of ColBERT model. We hope that they will support research, both within and outside of Mixedbread, while supporting a large range of real-world uses.

2 Creating a Suitable Dense Base Model

Previous work has demonstrated the importance of beginning ColBERT trainings from a suitably "warmed-up" model, with considerably better results obtained when training from a dense embedding model rather than initializing training from scratch [4,1], outperforming even those obtained by further training an existing ColBERT model [3,25]. We believe this effect to be due to the fact that dense embedding models now routinely undergo a long, distantly-supervised contrastive alignment training phase [31] before being fine-tuned on high quality data, which is not commonly done for ColBERT models[3].

In light of this, we first set out to create a suitable dense backbone, at our target model sizes: 32 and 17 million parameters. We use the Ettin [35] encoder

[3] As the aim of this work is to create a suitable backbone to identify the effect of individual modifications, we leave the exploration of a ColBERT-specific contrastive warm-up phase to future work, but believe it holds strong potential for further improvements.

models as starting models, which are a replication of the ModernBERT training recipe across various model sizes [33].

2.1 Contrastive Pre-Training

We follow the standardised recipe for our contrastive pre-training phase, as is now commonly adopted by the large majority of embedding models [31,19,16]. Effectively, this phase consists in leveraging many open datasets during which we have approximate queries that can be mapped to documents that are at least somewhat semantically related to them. In practice, this takes many different forms: forum posts with their title acting as a query, QA pairs extracted from common websites, etc. This training is done with a large batch size, facilitated by GradCache [9], and resulting in a better embedding alignment.

For this section, we used the contrastor training framework, which was used in the training of the Nomic embeddings models [21]. We used a common selection of pre-training datasets, presented in Table 2.1. We follow the work done on mxbai-embedding-large [12] and train sequentially, that is, one dataset at a time, rather than all at once, which we empirically found to result in better performance. A similar form of this effect was described in the snowflake-arctic embeddings tech report [19], where stratification of training examples by origin dataset yielded superior results.

Table 1. Datasets used for contrastive pretraining.

Dataset	**Size (rows)**
synthetic datasets	2.65M
nomic-embed-unsupervised-data[4]	172.8M
bge-m3-data	1.57M
cornstack (subsampled, 8% total)	20M
Total	197M

Interestingly, this pre-training phase highlighted an effect that appears common to all the ModernBERT and Ettin models: a higher learning rate is needed to reach satisfying results when compared to previous backbone encoder models. This effect was first described in the ModernBERT paper, where hyperparameter sweeps revealed that a considerably higher learning rate was necessary for ModernBERT to outperform previous encoders on common retrieval tasks [33]. Table 2 shows the NanoBEIR NDCG@10 of multiple training runs with different learning rates.

2.2 Fine-tuning

Subsequently, we move on to the next step of dense embedding pre-training: supervised fine-tuning on higher quality data, with mined hard negatives [37]. The mining of hard negatives is a key factor in training embedding models, as

Table 2. Performance of two models post-contrastive training with varying learning rates (NDCG@10 on NanoBEIR)

Model	Learning Rate	Batch Size	NDCG@10
17M	3.5e-04	24576	0.493
17M	6.0e-04	24576	**0.523**
32M	2.8e-04	12288	0.543
32M	5.0e-04	12288	**0.559**

it helps provide stronger "counter-examples": for every example that is relevant to the query, we also provide the models with examples that are not. When using solely random negatives, this task becomes trivial: a completely unrelated negative will rapidly only reach very low similarity scores, and stop meaningfully contributing to learning. This also dulls the learning process of what a "match" looks like: if the negative examples are always completely unrelated, then the model does not have to work as hard to learn what makes a document truly relevant. Hard negative mining attempts to solve this problem by gathering a set of harder negatives that look more similar to the positive document.

This ensures that the model learns to accurately represent details that differentiate relatively similar documents, rather than just general topics. On the other hand, negatives that are too hard can also be harmful to the learning process: if the model only ever sees negative examples that are "almost-positives", then it might fail to learn good high-level representations. Moreover, negatives that are too hard carry a high false-negative rate: most datasets commonly used for retrieval have sparse labels, and it is highly likely that a lot of the highest-scoring negative documents could actually be positives.

As such, crafting a good mix of negatives is important. We follow NV-Embedv2 [26] in our mining process, where we used Qwen3-Embedding-8B to mine hard negatives and set the threshold to 0.95. To learn on various negative hardness, we also mixed the data with 35% of BM-25 mined and 30% of randomly mined documents. We mine negative for the de-facto standard set of training datasets for retrieval fine-tuning: MSMARCO, NQ, HotPotQA and PubMed.

Table 3. Performance comparison of Mxbai Edge models (Dense) with and without finetuning (NDCG@10).

Model	NDCG@10
Mxbai Edge 17M (Dense, non-FT)	0.523
Mxbai Edge 17M (Dense, FT)	**0.556**
Mxbai Edge 32M (Dense, non-FT)	0.559
Mxbai Edge 32M (Dense, FT)	**0.576**

We adopt the AnglE training loss [15] for this fine-tuning step, using the AnglE codebase[5]. We train on a selection of common datasets, once again seeking to follow standard practices while avoiding over-contamination in relation to frequent benchmarks.

2.3 "Stella-style" Distillation

Finally, we add a third stage to our model pre-training, which is inspired by the Stella [38] model family. Stella, in addition to more commonplace retrieval training, introduces **embedding space distillation**: it generates embeddings for queries and documents using a strong teacher, such as LLM-based embedding models, and designs a teaching process in which various distance-based losses are used to minimise the distance between the embeddings produced by the student model and its teachers. The resulting models, the Stella and Jasper embedding families, are extremely strong embedding models at their respective sizes, and have been frequently demonstrated to reach very strong out-of-domain performance [17].

As part of our training, we initially employed the partial codebase released by the Stella authors[6], but found the full multi-step process difficult produce, yielding poorly performing models, with fluctuating performance and extremely high sensitivity to hyperparameters. Following work such as LEAF [30], we opted to simplify the distillation loss to a simple L2 loss:

$$\mathcal{L}_2(y_i, \hat{y}_i) = \sum_{j=1}^{d} (y_{ij} - \hat{y}_{ij})^2 \tag{1}$$

which attempts to minimize the distance between our student's vectors and the teacher vectors.

As this step relies purely on embedding space distillation, there is no need to leverage retrieval datasets, since the relationship between queries and documents is not used during this stage. However, it has previously been highlighted that having a variety of inputs corresponding to common retrieval uses [30], especially in terms of input lengths (e.g. longer documents and short queries), is helpful to improve performance. We thus sample many documents from various mixed sources and queries from large retrieval datasets, with a detailed data mix provided in Appendix A.

We used StellaV5 1.5B as our teacher model, with an output dimension of 1024. We found that using higher teacher dimension embeddings resulted in decreasing performance, likely due to the vast dimension difference between our student models and the target sizes, while lower dimensions such as 768 resulted in mildly diminished results. To ensure our models' dimensions matched the teacher ones for distillation, we employed a 2-layer feedforward projection with a SiLU [7](a.k.a Swish [23]) activation.

[5] `https://github.com/SeanLee97/AnglE`
[6] `https://github.com/NovaSearch-Team/RAG-Retrieval`

Table 4. Average NDCG@10 on NanoBEIR for dense Mxbai Edge variants.

Model	Avg. NDCG@10
Mxbai Edge 17M (Dense, FT)	0.556
Mxbai Edge 17M (Dense, Distill)	**0.567**
Mxbai Edge 32M (Dense, FT)	0.576
Mxbai Edge 32M (Dense, Distill)	**0.626**

Table 4 shows the NanoBEIR NDCG@10 results between the post-finetuning and post-distillation variants of both model sizes. It shows that, despite our streamlined process, this step results in performance gains[7], but unevenly distributed across model sizes. While the 32M variant heavily benefits from this step, the gains on the 17M model are more modest. We theorise that this might be due to the streamlined distillation loss we used struggling to bridge large dimensionality gaps compared to the original, more complex Stella loss mix, but do not explore this effect further.

3 ColBERT Training

Finally, we apply our final training stage for the ColBERT models. We run a series of ablations in order to create a strong baseline with this model, which will be able to support both real-world edge use cases and subsequent research uses satisfyingly.

We detail our training setting in the subsequent sections about our ablation work. For training data, we restrict ourselves to MSMARCO, so as to ensure that better data does not obscure the impact of training modifications (further details in Section 3.1), and use 16-way training tuples, where each query is associated with a positive example and 15 negative examples, all with a teacher score. We use a batch size of 128 and KL-Div loss with normalized scores [3], except when otherwise specified. All experiments are performed using the PyLate [2] framework.

3.1 Data

We experimented with various training datasets, comparing the MSMARCO [20] RLHN [29] set scored with Qwen3-Reranker [39] as a teacher to the triplets used by answerai-colbert-small [4] and GTE-ModernColBERT [1], with scores generated by BGE-Gemma2 Reranker[8] [14] to score a small subset of MSMARCO training tuples and comparing min-max normalized and unnormalized teacher scores [3].

[7] with the 32M variant reaching performance that is competitive with many state-of-the-art small embedding models.

[8] `https://huggingface.co/BAAI/bge-reranker-v2-gemma`

Table 5. Effect of teachers used for distillation.

Teacher	NDCG@10
Qwen3-8B (no norm)	0.5991
Qwen3-8B (minmax norm)	0.5854
BGE-M3	**0.6286**

We present a comparison of these training methods in Table 5. Surprisingly, whether normalized or unnormalized, Qwen3-Reranker is consistently outperformed as a teacher by the older BGE-Gemma2. Upon further analysis, it appears that on common retrieval benchmarks, the scores generated by Qwen3-Reranker are extremely skewed towards the extremes, with very scores outside of the [0.99,1] range for positives and [0, 0.01] range for negatives. We believe that this might be indicative of overfitting from the reranker, resulting in poor distributions to use for distillation. Our attempts at using both large and very small temperatures did not significantly change performance.

3.2 Ablations

We performed various ablations in order to understand the impact of certain parameters on model performance. To avoid overfitting, hyperparemeter ablations (optimizer, distillation impact, learning rate and projection dim) were evaluated on 5 NanoBEIR subset, so as to provide a good performance indicator without being exposed to the full BEIR sets. The selected subsets are high-quality search dataset MSMARCO (in-domain), SciFact (OOD), FiQA (OOD), NQ (OOD), and NFCorpus (OOD). Final ablations on projection layers and casing were evaluated on all of NanoBEIR.

Optimizers We benchmarked both AdamW [18] and Muon [10] across a range of learning rates with a fixed batch size. We present the results of these ablations in Table 6. Our results indicate that even with limited experiments and the relatively small batch size that is commonly employed to train late-interaction models, Muon appears to be a strong optimizer for ColBERT model training.

Impact of Stella-style Distillation In Table 7, we show a comparison of running trainings on the dense embedding result model resulting from our fine-tuning stage vs the model resulting from our distillation stage. Our results clearly show that Stella-style distillation improves performance of the resulting ColBERT model, even when projection heads are discarded to only retain the backbone model.

Projection Dimension The projection dimension used by ColBERT models is traditionally set to 128, after the one used by the original ColBERT and ColBERTv2 [11,25] models, and this dimension has shown good performance in

Table 6. Comparison of model performance across optimizers and learning rates. NDCG@10 is the average NDCG@10 score across the 4 ablation datasets.

Optimizer	NDCG@10
AdamW	
1e-4	0.5911
5e-5	0.5780
8e-5	0.5923
Muon	
1e-4 (AdamW 8e-5)	0.5718
3e-4 (AdamW 8e-5)	0.5604
5e-4 (AdamW 8e-5)	0.5862
1e-3 (AdamW 8e-5)	**0.5985**
3e-3 (AdamW 8e-5)	0.5748

Table 7. ColBERT performs better on a base embedding model trained on Stella-style distillation.

Base Model Variant	NDCG@10
32M model (fine-tuned only)	0.5771
32M model (with Stella-style distillation)	**0.5911**

both text and multimodal settings [8]. As of right now, the current state-of-the-art for smaller ColBERT models uses a projection dimension of 96 [4]. In effect, the final projection dimension is largely defined in an arbitrary way, despite the large consequences it has in terms of both storage requirements and scoring speed.

Table 8. Effect of projection dimension on NDCG@10 (Muon 1e-3, AdamW 8e-5) on the 32m model, using 20% training data.

Projection Dimension	NDCG@10
96	**0.5991**
64	0.5985
48	0.5967
32	0.5772
24	0.5423
16	0.5126

In Table 8, we present the results of ablating a large range of projection dimensions, from 16 to 96. We show that lower dimensions hold up performance surprisingly well. Indeed, the performance decrease on NanoBEIR is very mild

until a dimension of 48, but subsequently considerably degrades at projection dimensions of 32 and below.

Projection Layers In a recent study, we demonstrated that the use of more complex projection layers outperformed the single-layer linear projection that is ubiquitous in ColBERT models [5]. As part of this work, we experiment with the best variant proposed in our previous study, using a 2-layer feedforward network with an upscaled intermediate dimension and a residual connection, and compare it to a model trained with the "normal" ColBERT projection.

Table 9. Performance of different projection heads on the 17m model, under matched training hyperparameters, on full data.

Projection	NDCG@10
2-layer FFN	**0.6405**
Linear Projection	0.6286

We present the results of this comparison on the 17m parameter model variant in Table 9. As this experiment came later in our training process, results are reported as full NanoBEIR NDCG@10 rather the previously defined ablation sets. Our results that the use of better projection layers contributes positively to performance. While we do not perform significance testing due to the low number of evaluated checkpoints, we note that we reproduced this effect across a range of training seeds, with no single-layer linear projection checkpoint coming within less than 1 NDCG@10 of the 2-layer projection checkpoints.

Casing Virtually all embedding models we consider "previous-generation" either use BERT-BASE-UNCASED [6] as their backbone, or models which were largely inspired by it. These encoders are all case-insensitive, meaning that all input text is lower-cased before being tokenized.

On the other hand, virtually all Large Language Models employ some form of casing, which ModernBERT [33], and thus subsequently Ettin [35], also adopts.

The impact this tokenization change has, if any, has not yet been studied in detail. We decided to conduct an ablation in this sense, for which we present the results in Table 10.

Our results demonstrate an interesting phenomenon: while there appears to be no significant difference at the 32M parameter scale, the results of the 17M model variant are significantly improved by lower-casing. Across random seeds, we also observed that lower-casing consistently reached stronger performance on the 17M model, but no discernible trend emerged at the 32M scale.

As with other ablations, we do not further attempt to understand the underlying mechanism, but theorise that the limited embedding dimensions and parameter count of the 17M model means that it benefits disproportionately from the learning simplification that lower-casing provides.

Table 10. NanoBEIR NDCG@10 comparison with and without lower-casing as a preprocessing step, with all other hyperparameters kept equal and training on the full data.

Optimizer	NDCG@10
32M	
Lower-casing	**0.6520**
No lower-casing	**0.6519**
17M	
Lower-casing	**0.6405**
No lower-casing	0.6317

4 Results

Table 11. BEIR benchmark results (NDCG@10). Columns show the BEIR average and sampled tasks: MSMARCO, SciFact, Touche2020, FiQA, TREC-COVID, NQ, and DBPedia. Results in bold indicate best result for the weight class. The best results for size class are in **bold**. The complete table in Appendix B.

Model	AVG	MSMARCO	SF	Touche	FiQA	COVID	NQ	DBP
>100M parameters								
GTE-ModernColBERT-v1	**0.547**	0.453	**0.763**	**0.312**	**0.453**	**0.836**	**0.618**	**0.480**
ColBERTv2	0.488	**0.456**	0.693	0.263	0.356	0.733	0.562	0.446
<35M parameters								
mxbai-edge-colbert-v0-32m	0.521	**0.450**	**0.740**	**0.313**	0.390	0.775	**0.600**	0.455
answerai-colbert-small-v1	**0.534**	0.434	**0.740**	0.250	**0.410**	**0.831**	0.594	**0.464**
bge-small-en-v1.5	0.517	0.408	0.713	0.260	0.403	0.759	0.502	0.400
snowflake-s	0.520	0.402	0.722	0.235	0.407	0.801	0.509	0.410
<25M parameters								
mxbai-edge-colbert-v0-17m	**0.490**	**0.416**	**0.719**	**0.316**	0.326	**0.713**	**0.551**	**0.410**
colbert-muvera-micro	0.394	0.364	0.662	0.251	0.254	0.561	0.386	0.332
all-MiniLM-L6-v2	0.419	0.365	0.645	0.169	**0.369**	0.472	0.439	0.323

In Table 11, we present the result of our model on a selected range of BEIR [28] datasets and their average, while Table 12, we present the result of our model on LongEmbed task [41].

On the BEIR datasets, we note that our models are overall strong performers. While outperformed on short-context average by the previous small-scale state-of-the-art, answerai-colbert-small, they reach strong performance across

the board. Particularly noteworthy is that mxbai-edge-colbert-v0-17m, a 17 Million parameter model, outperforms the still-widely-used ColBERTv2, despite having less than 1/6th of the parameters and a projection dimension set to just 48, a third of ColBERTv2's 128. They do so with remarkable efficiency, especially as context length increases, thanks to their ModernBERT-based backbone.

Table 12. Detailed LongEmbed benchmark performance. Context length is set to 4k and 32k context variants for models supporting it. Otherwise, it is set to the model's maximum sequence length (8k for granite-embeddings and 512 for others). Best results for size class are in **bold**, best overall results are underlined. Models with more parameters than their size class but added for completeness are in *italics*.

Model	AVG	NarrQA	QMSum	Wiki	SummScr.	Needle	Passkey
>100M parameters							
GTE-ModernColBERT-v1 (32k)	**0.898**	**0.780**	**0.737**	**0.999**	**0.953**	**0.950**	**0.970**
GTE-ModernColBERT-v1 (4k)	0.809	0.530	0.528	0.931	0.947	**0.950**	**0.970**
granite-embedding-english-r2[9]	0.656	0.479	0.416	0.859	0.937	0.430	0.818
ColBERTv2	0.428	0.287	0.254	0.648	0.686	0.330	0.365
<50M parameters							
mxbai-edge-colbert-v0-32m (32k)	**0.849**	**0.585**	**0.698**	**0.993**	**0.910**	**0.915**	**0.990**
mxbai-edge-colbert-v0-32m (4k)	0.783	0.444	0.508	0.930	0.909	**0.915**	**0.990**
granite-embedding-small-english-r2[10]	0.637	0.413	0.365	0.799	0.899	0.550	0.798
answerai-colbert-small-v1	0.441	0.266	0.272	0.645	0.735	0.338	0.388
bge-small-en-v1.5	0.312	0.220	0.208	0.430	0.532	0.263	0.218
snowflake-arctic-embed-s	0.356	0.177	0.230	0.411	0.643	0.283	0.390
<25M parameters							
mxbai-edge-colbert-v0-17m (32k)	**0.847**	**0.621**	**0.733**	**0.977**	**0.943**	**0.950**	**0.858**
mxbai-edge-colbert-v0-17m (4k)	0.776	0.437	0.566	0.909	0.935	**0.950**	**0.858**
all-MiniLM-L6-v2	0.298	0.183	0.163	0.463	0.548	0.200	0.233
colbert-muvera-micro	0.405	0.230	0.244	0.566	0.689	0.318	0.385

On long-context evaluations, our models reach very strong performance, only outperformed again by the larger GTE-ModernColBERT. We show that both of our models are extremely strong performer, only outperformed by the larger GTE-ModernColBERT. As expected, models based on more modern architecture, capable of handling longer context lengths, are the only competitive models on this task, with previous methods unable to process longer documents efficiently and resorting to truncation, thus greatly reducing their performance.

[9] 149M parameter model. Results are from the MTEB leaderboard.
[10] 49M parameter model. Results are from the MTEB leaderboard.

Particularly notably, even our 17M parameter variant outperforms the current <1B parameter single-vector retrieval state-of-the-art[11] on LongEmbed tasks, such as granite-embedding-r2, by almost 20NDCG@10 points. Note that Needle and Passkey are computed on NDCG@1 and are calculated by taking the average of all lengths.

Interestingly, we note, similarly to [1] that despite being based on a model with a native 8,000 context window, mxbai-edge-colbert's models are capable of handling 32k sequence lengths and observe performance gains from it, despite our retrieval training using documents truncated to 220 tokens.

Finally, we note that the low parameter count of our models, in combination with their highly-efficient architecture, make them particularly suitable for reranking task. This is especially true for longer chunks, as there currently does not exist any re-ranker able to reach similarly strong performance while running with low latencies on CPU for long document reranking.

Table 13. Relative performance and efficiency comparisons of small ColBERT models on NanoBEIR, with ColBERTv2 as a reference. CPU and GPU refer to runtimes as the average of 10 runs on each hardware type. Mem. is RAM requirements, in MB, for storing 10,000 300 token document representations in fp16. LoCo. stands for long-context support. Dim. is the projection dimension of each model. Best values are in **bold**, best values while outperforming ColBERTv2 on retrieval are underlined.

Model	Params	Dim.	NDCG@10	LoCo	GPU	CPU	Mem.
ColBERTv2	130M	128	0.6198	–	81s	1540s	732
answerai-colbert-small-v1	33M	96	**0.6545**	–	59s	621s	549
colbert-muvera-micro	4M	128	0.5599	–	**45s**	**88s**	732
mxbai-edge-colbert-v0-17m	17M	48	0.6405	✓	51s	487s	**275**
mxbai-edge-colbert-v0-32m	32M	64	0.6520	✓	55s	589s	366

Efficiency comparison with other small ColBERTs Table 13 shows the relative performance of our 17M parameter edge ColBERT against other commonly used small ColBERTs, along with efficiency comparisons. For ease of rapid evaluation, we report overall NanoBEIR NDCG@10 scores, long-context support, projection dimension (a very important factor for edge use cases), mean runtime of 10 NanoBEIR evaluation runs on both GPU with a single RTX 4090 and CPU, representing the encoding of around 67,000 documents, as well as 650 queries and as many searches and scoring steps.

We also report the memory usage required to store the 16-bit vector representations of 10,000 300-tokens documents stored, a direct factor of the projection dimension, to provide a brief overview of the suitability for various in-memory encoding usages.

[11] According to the MTEB Leaderboard as of October 2025

5 Conclusion

We introduce v0 of the Mxbai-edge-ColBERT model family. These models represent the first small ColBERT model to fully benefit from a modern architecture, with long-context support and all the efficiency improvements introduced by the ModernBERT [33] generation of encoder models.

Our intent with these models is two-fold: our main aim is for them to provide a suitable testbed for future experiments and distillation of our research on larger-scale models, as well as to serve as a strong performance predictor experiments, following scaling laws. Our second aim to support a large range of on-device use-cases, be it local RAG projects or extremely efficient re-ranking on both CPU and GPU.

Mxbai-edge-ColBERT-v0, at both model sizes, reach strong performance on a variety of datasets. Notably, the 17M parameter variant outperforms ColBERTv2 with an order-of-magnitude fewer parameters, and with vector storage and scoring-time compute requirements reduced by two thirds.

We fully intend to continue to upgrade these models with future developments and are looking forward to seeing them used in the real-world.

References

1. Chaffin, A.: Gte-moderncolbert (2025), https://huggingface.co/lightonai/GTE-ModernColBERT-v1
2. Chaffin, A., Sourty, R.: Pylate: Flexible training and retrieval for late interaction models. arXiv preprint arXiv:2508.03555, to be published at CIKM 2025 (2025)
3. Clavié, B.: Jacolbertv2. 5: Optimising multi-vector retrievers to create state-of-the-art japanese retrievers with constrained resources. Journal of Natural Language Processing **32**(1), 176–218 (2025)
4. Clavié, B.: Small but mighty: Introducing answerai-colbert-small (August 2024), https://www.answer.ai/posts/2024-08-13-small-but-mighty-colbert.html
5. Clavié, B., Lee, S., Takehi, R., Shakir, A., Kato, M.P.: Simple projection variants improve colbert performance (2025), https://arxiv.org/abs/2510.12327
6. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). pp. 4171–4186 (2019)
7. Elfwing, S., Uchibe, E., Doya, K.: Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. Neural networks **107**, 3–11 (2018)
8. Faysse, M., Sibille, H., Wu, T., Omrani, B., Viaud, G., HUDELOT, C., Colombo, P.: Colpali: Efficient document retrieval with vision language models. In: The Thirteenth International Conference on Learning Representations (2025), https://openreview.net/forum?id=ogjBpZ8uSi
9. Gao, L., Zhang, Y., Han, J., Callan, J.: Scaling deep contrastive learning batch size under memory limited setup. In: Proceedings of the 6th Workshop on Representation Learning for NLP (2021)

10. Jordan, K., Jin, Y., Boza, V., You, J., Cesista, F., Newhouse, L., Bernstein, J.: Muon: An optimizer for hidden layers in neural networks (2024), https://kellerjordan.github.io/posts/muon/
11. Khattab, O., Zaharia, M.: Colbert: Efficient and effective passage search via contextualized late interaction over bert. In: Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval. pp. 39–48 (2020)
12. Lee, S., Shakir, A., Koenig, D., Lipp, J.: Open source strikes bread-new fluffy embeddings model (2024). URL https://www. mixedbread. ai/blog/mxbai-embed-large-v1 (2024)
13. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.t., Rocktäschel, T., et al.: Retrieval-augmented generation for knowledge-intensive nlp tasks. Advances in Neural Information Processing Systems **33**, 9459–9474 (2020)
14. Li, C., Liu, Z., Xiao, S., Shao, Y.: Making large language models a better foundation for dense retrieval (2023)
15. Li, X., Li, J.: AnglE-optimized text embeddings. arXiv preprint cs.CL arXiv:2309.12871 (2023)
16. Li, Z., Zhang, X., Zhang, Y., Long, D., Xie, P., Zhang, M.: Towards general text embeddings with multi-stage contrastive learning. arXiv preprint arXiv:2308.03281 (2023)
17. Liu, F., Enevoldsen, K.C., Solomatin, R., Samoed, T., Chung, I., Aarsen, T., Fődi, Z.: Introducing rteb: A new standard for retrieval evaluation. Hugging Face Blog (Oct 2025), https://huggingface.co/blog/rteb
18. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017)
19. Merrick, L., Xu, D., Nuti, G., Campos, D.: Arctic-embed: Scalable, efficient, and accurate text embedding models. arXiv preprint arXiv:2405.05374 (2024)
20. Nguyen, T., Rosenberg, M., Song, X., Gao, J., Tiwary, S., Majumder, R., Deng, L.: Ms marco: A human-generated machine reading comprehension dataset (2016)
21. Nussbaum, Z., Morris, J.X., Duderstadt, B., Mulyar, A.: Nomic embed: Training a reproducible long context text embedder (2024)
22. Penedo, G., Kydlíček, H., Lozhkov, A., Mitchell, M., Raffel, C.A., Von Werra, L., Wolf, T., et al.: The fineweb datasets: Decanting the web for the finest text data at scale. Advances in Neural Information Processing Systems **37**, 30811–30849 (2024)
23. Ramachandran, P., Zoph, B., Le, Q.V.: Searching for activation functions. arXiv preprint arXiv:1710.05941 (2017)
24. Santhanam, K., Khattab, O., Potts, C., Zaharia, M.: Plaid: an efficient engine for late interaction retrieval. In: Proceedings of the 31st ACM International Conference on Information & Knowledge Management. pp. 1747–1756 (2022)
25. Santhanam, K., Khattab, O., Saad-Falcon, J., Potts, C., Zaharia, M.: Colbertv2: Effective and efficient retrieval via lightweight late interaction. In: Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 3715–3734 (2022)
26. de Souza P. Moreira, G., Osmulski, R., Xu, M., Ak, R., Schifferer, B., Oldridge, E.: Nv-retriever: Improving text embedding models with effective hard-negative mining (2025), https://arxiv.org/abs/2407.15831
27. Teiletche, P., Macé, Q., Conti, M., Loison, A., Viaud, G., Colombo, P., Faysse, M.: Modernvbert: Towards smaller visual document retrievers (2025), https://arxiv.org/abs/2510.01149

28. Thakur, N., Reimers, N., Rücklé, A., Srivastava, A., Gurevych, I.: Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models. arXiv preprint arXiv:2104.08663 (2021)
29. Thakur, N., Zhang, C., Ma, X., Lin, J.: Fixing data that hurts performance: Cascading llms to relabel hard negatives for robust information retrieval (2025), https://arxiv.org/abs/2505.16967
30. Vujanic, R., Rueckstiess, T.: Leaf: Knowledge distillation of text embedding models with teacher-aligned representations (2025), https://arxiv.org/abs/2509.12539
31. Wang, L., Yang, N., Huang, X., Jiao, B., Yang, L., Jiang, D., Majumder, R., Wei, F.: Text embeddings by weakly-supervised contrastive pre-training. arXiv preprint arXiv:2212.03533 (2022)
32. Wang, W., Wei, F., Dong, L., Bao, H., Yang, N., Zhou, M.: Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. Advances in Neural Information Processing Systems **33**, 5776–5788 (2020)
33. Warner, B., Chaffin, A., Clavié, B., Weller, O., Hallström, O., Taghadouini, S., Gallagher, A., Biswas, R., Ladhak, F., Aarsen, T., Adams, G.T., Howard, J., Poli, I.: Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. In: Che, W., Nabende, J., Shutova, E., Pilehvar, M.T. (eds.) Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 2526–2547. Association for Computational Linguistics, Vienna, Austria (Jul 2025). https://doi.org/10.18653/v1/2025.acl-long.127, https://aclanthology.org/2025.acl-long.127/
34. Weller, O., Boratko, M., Naim, I., Lee, J.: On the theoretical limitations of embedding-based retrieval (2025), https://arxiv.org/abs/2508.21038
35. Weller, O., Ricci, K., Marone, M., Chaffin, A., Lawrie, D., Durme, B.V.: Seq vs seq: An open suite of paired encoders and decoders (2025), https://arxiv.org/abs/2507.11412
36. Yates, A., Nogueira, R., Lin, J.: Pretrained transformers for text ranking: Bert and beyond. In: Proceedings of the 14th ACM International Conference on web search and data mining. pp. 1154–1156 (2021)
37. Zhan, J., Mao, J., Liu, Y., Guo, J., Zhang, M., Ma, S.: Optimizing dense retrieval model training with hard negatives. In: Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval. pp. 1503–1512 (2021)
38. Zhang, D., Li, J., Zeng, Z., Wang, F.: Jasper and stella: distillation of sota embedding models (2025), https://arxiv.org/abs/2412.19048
39. Zhang, Y., Li, M., Long, D., Zhang, X., Lin, H., Yang, B., Xie, P., Yang, A., Liu, D., Lin, J., Huang, F., Zhou, J.: Qwen3 embedding: Advancing text embedding and reranking through foundation models (2025), https://arxiv.org/abs/2506.05176
40. Zhou, F., Wang, Z., Liu, Q., Li, J., Liu, P.: Programming every example: Lifting pre-training data quality like experts at scale. arXiv preprint arXiv:2409.17115 (2024)
41. Zhu, D., Wang, L., Yang, N., Song, Y., Wu, W., Wei, F., Li, S.: Longembed: Extending embedding models for long context retrieval (2024), https://arxiv.org/abs/2404.12096

A Distillation Data

The data mix for the distillation stage is provided in Tables 14 and 15.

Table 14. Queries used for distillation

Dataset	Size (rows)
msmarco	510k
amazon_qa	475k
nq	175k
triviaqa	70k
pubmed	67k
arxiv	50k
cornstk	50k
lotte	25k
medqa	13k
mldr	12.2k
Total	**1.45M**

Table 15. Passages used for distillation

Dataset	Size (rows)
DCLM-Pro	1.59M
english_words	742k
fineweb	665k
dclm_sent	400k
ccnews	370k
stack	185k
ettin_tokens	50k
Total	**4.00M**

DCLM-Pro [40] and FineWeb [22] are full documents randomly from their respective datasets, while dclm_sent is comprised of individual DCLM-Pro documents broken down into individual sentences, to create more varied small-length inputs, again following Stella [38]. ettin_tokens and english_words were added during the course of this study following the release of LEAF [30], which used a similar method to improve trianing. ettin_tokens is a dataset comprised of very short input, where each document is a single token from our model's tokenizer, while english_words is a large collection of English words along with a definition generated by Gemini 2.0 Flash.

B Full BEIR Results

We show the full BEIR results in Tables 16 and 17.

Table 16. BEIR benchmark (Part A): AVG and (Touche2020, NQ, MSMARCO, SciFact, FiQA2018, NFCorpus, ArguAna). Scores are NDCG@10.

Model	AVG	Touche2020	NQ	MSMARCO	SciFact	FiQA2018	NFCorpus	ArguAna
>100M parameters								
GTE-ModernColBERT-v1	**0.547**	**0.312**	**0.618**	0.453	**0.763**	**0.453**	**0.379**	**0.485**
colbertv2	0.488	0.263	0.562	**0.456**	0.693	0.356	0.338	0.463
<35M parameters								
mxbai-edge-colbert-v0-32m	0.521	**0.313**	**0.600**	**0.450**	**0.740**	0.390	0.362	0.454
answerai-colbert-small-v1	**0.533**	0.250	0.594	0.434	**0.740**	**0.410**	**0.369**	**0.468**
bge-small-en-v1.5	0.517	0.260	0.502	0.408	0.713	0.403	0.349	0.331
snowflake-s	0.520	0.235	0.509	0.402	0.722	0.407	0.324	0.339
<25M parameters								
mxbai-edge-colbert-v0-17m	**0.490**	**0.316**	**0.551**	**0.416**	**0.719**	0.326	**0.352**	**0.464**
colbert-muvera-micro	0.394	0.251	0.386	0.364	0.662	0.254	0.321	0.303
all-MiniLM-L6-v2	0.419	0.169	0.439	0.365	0.645	**0.369**	0.314	0.331

Table 17. BEIR benchmark (Part B): Rest of the tasks (QuoraRetrieval, SCIDOCS, TRECCOVID, ClimateFEVER, HotpotQA, DBPedia, CQADupstack, FEVER). Scores are NDCG@10.

Model	QuoraRetrieval	SCIDOCS	TRECCOVID	ClimateFEVER	HotpotQA	DBPedia	CQADupstack	FEVER
>100M parameters								
GTE-ModernColBERT-v1	**0.866**	**0.191**	**0.836**	**0.306**	**0.773**	**0.480**	**0.410**	**0.874**
colbertv2	0.852	0.154	0.733	0.176	0.667	0.446	0.378	0.785
<35M parameters								
mxbai-edge-colbert-v0-32m	0.863	0.170	0.775	0.290	0.734	0.455	0.388	0.826
answerai-colbert-small-v1	0.879	0.187	**0.831**	**0.328**	**0.769**	**0.464**	0.394	**0.887**
bge-small-en-v1.5	**0.887**	**0.198**	0.759	0.253	0.699	0.400	0.391	0.866
snowflake-s	0.884	0.218	0.801	0.352	0.665	0.410	**0.397**	0.871
<25M parameters								
mxbai-edge-colbert-v0-17m	0.839	0.169	**0.713**	**0.224**	**0.713**	**0.410**	0.356	**0.784**
colbert-muvera-micro	0.764	0.123	0.561	0.115	0.528	0.332	0.313	0.637
all-MiniLM-L6-v2	**0.876**	**0.217**	0.472	0.203	0.465	0.323	**0.412**	0.519