SSE: A Parameter-Efficient Training Recipe for Static Retrieval Embeddings

Rikka Botan[1]
[1]Independent Researcher, Japan
Contact: RikkaBotan@outlook.jp | https://rikka-botan.github.io

Abstract: Static retrieval embeddings provide fast inference but often trail contextual encoders in accuracy. We present SSE (Stable Static Embedding), a complete training recipe that combines contrastive ranking, Matryoshka prefix supervision, a 15-source data mixture, large-batch one-epoch optimization, and a lightweight post-pooling transform. The released checkpoint has 15.6M parameters, approximately half the 31.3M parameters of the prior static reference. Its 256-dimensional prefix reaches NanoBEIR (English) nDCG@10 of 0.5036, matching the reference's native 1024-dimensional score of 0.5030 with one quarter of the vector storage; the full 512-dimensional representation reaches 0.5159. The model encodes 119,832 queries per second on CPU, has the highest measured quality within the throughput-comparable static family, and lies on the measured speed–quality Pareto frontier. These results support SSE as a parameter- and storage-efficient recipe for static dense retrieval.

Keywords: static retrieval embeddings; parameter efficiency; dense retrieval; Matryoshka representation learning; NanoBEIR

1. INTRODUCTION

Dense vector representations have become a fundamental component of modern information retrieval and retrieval-augmented language systems. However, a significant trade-off exists between accuracy and efficiency in current architectures. While contextual embedding models based on transformer architectures achieve strong semantic performance, their computational cost during inference remains a major bottleneck for large-scale and latency-sensitive applications. In contrast, static embedding models offer extremely fast inference and low memory consumption due to their simple and deterministic structure. These properties make them particularly attractive for large-scale search, recommendation, and real-time retrieval systems. Consequently, there is a widening gap between static embeddings and contextual encoders: the former provide speed but often lack accuracy, while the latter provide accuracy but are computationally expensive. Bridging this gap requires new techniques that enhance the representational quality of embedding models without introducing significant computational overhead.

The development of word representations has evolved significantly to address these needs since the pioneering work on Word2Vec [1], which established distributed word embeddings based on distributional hypotheses. This was followed by GloVe [2], which introduced global co-occurrence statistics for more stable learning, and fastText [3], which incorporated subword information to improve robustness for rare words. These models enabled practical deployment in large-scale systems through their fixed token representations and lightweight composition functions. Recent releases include `static-similarity-mrl-multilingual-v1` and `static-retrieval-mrl-en-v1` [4, 5]. They show that static embeddings can achieve practical retrieval performance while running 400 times faster than transformer-based models [6].

Improving static retrieval is not only an architectural question. Quality depends jointly on data composition, contrastive supervision, prefix objectives, the post-pooling map, batch scale, and the optimization schedule. Because these choices interact, the primary object of evaluation is the complete training recipe. Component-level controls are reported separately in Appendix A.

In this work, we present SSE (Stable Static Embedding), an end-to-end training recipe for lookup-and-pool retrievers. It combines a fixed tokenizer and 512-dimensional embedding table, exact mean pooling, a lightweight learned post-pooling map, MultipleNegativesRankingLoss, Matryoshka supervision over five prefix widths, a 15-source data mixture, a large effective batch, and one epoch of training. The central result is dimensional and parameter efficiency: the 256-dimensional prefix reaches 0.5036 on NanoBEIR (English), matching the prior static reference's native 1024-dimensional score of 0.5030 with one quarter of the vector storage. The released checkpoint has 15.6M parameters, approximately half the reference's 31.3M, while its full 512-dimensional representation reaches 0.5159.

Main contributions.

1. We introduce a parameter-efficient static-retrieval training recipe whose 15.6M-parameter checkpoint is approximately half the size of the 31.3M-parameter reference; its 256-dimensional prefix matches the native 1024-dimensional reference with one quarter of the vector storage.
2. We show that the post-pooling gain arises from learned representation reorganization rather than rank reduction. The near-full-rank `sepdyt_no_prune` variant matches `sepdyt`, removing the learned map reduces nDCG@10 by 0.0098, and a fixed rank mask does not reproduce the gain. The lower-rank spectrum is an emergent signature of the improved representation.
3. We re-measure CPU throughput using per-model batch sweeps, repeated confidence intervals, and interleaved static-model runs, showing that the released model lies on the measured speed–quality Pareto frontier.

The released model is available at https://huggingface.co/RikkaBotan/stable-static-embedding-fast-retrieval-mrl-en-v2 under Apache-2.0. The reference implementation, quantization tools, and evaluation artifacts are available at https://github.com/Rikka-Botan/Stable-Static-Embedding-Models under the MIT License.

Received date: Aug. 30 2026
Accepted date: Sept. 15 2026

DOI: 10.63211/j.p.26.204501
ISSN 2760-8646 (Online) ICAITD

2. RELATED WORK

Static token representations. Word2Vec, GloVe, and fastText established scalable token representations based on local prediction, global co-occurrence, and subword composition [1–3]. Sentence-level static encoders aggregate token vectors by sum or mean, often with frequency-based weighting. The SIF baseline of Arora et al. [7] showed that these weighted averages, combined with common-component removal, provide a strong baseline. SSE follows the same architectural family—a lookup table plus a parameter-free pooling operation—and combines contrastive training, nested prefix supervision, and a small learned post-pooling map in one recipe.

Trained static retrievers. Recent Sentence Transformers recipes train the embedding table directly for retrieval with contrastive supervision and report 100–400× CPU speedups over common contextual encoders [6]. The reference releases are `static-retrieval-mrl-en-v1` and `static-similarity-mrl-multilingual-v1` [4, 5]. Another approach distills a contextual encoder into a static table. Model2Vec and the `potion` series use output distillation of sentence encoders [8]. The models range from `potion-base-2M` to `potion-base-32M` [9] and include the multilingual `potion-multilingual-128M` [10]. They also include `M2V_base_glove`, which applies the same distillation to GloVe vectors [11]. These systems and an untrained mean-pooled GloVe table [12] form the static family used in our quality and speed comparison (Section 5.3). The GloVe table provides a baseline before contrastive training.

Lexical retrieval. BM25 is a strong sparse lexical retrieval baseline based on term frequency, inverse document frequency, and document-length normalization [13]. We evaluate BM25 on the same NanoBEIR queries, corpora, and relevance judgments as the dense models. Because BM25 scores an inverted index rather than producing fixed-dimensional vectors, its quality is reported separately from embedding dimension, checkpoint parameter count, and encoder-throughput comparisons.

Contrastive sentence encoders. Siamese encoders such as Sentence-BERT learn sentence vectors with contrastive supervision [14]. MiniLM-style distillation [15], weakly supervised contrastive pre-training in E5 [16, 17], and the BGE/C-Pack resources [18] provide the high-accuracy reference points in our comparison. Cross-architecture distillation [19] shows how much of that quality transfers to cheaper rankers. The representational geometry of such encoders is commonly analysed through alignment and uniformity [20] and is known to be anisotropic [21]. These findings motivate diagnostic analysis of the geometry learned by static recipes.

Isotropy correction and post-processing. Prior work has modified embedding geometry by removing the mean and dominant principal directions in All-but-the-Top [22], mapping sentence embeddings to an isotropic Gaussian with BERT-flow [23], and applying a closed-form whitening transform [24]. SimCSE further relates contrastive learning to alignment and uniformity [25]. These approaches show that post-processing and training objectives can both affect sentence-embedding geometry; our analysis instead tests the rank behavior produced within a static training recipe.

Normalization and Dynamic Tanh. DyT replaces normalization layers in Transformers with $\tanh(\alpha x)$ followed by a per-coordinate affine map, where α is a single shared scalar [26]. Separable DyT makes all three quantities per-coordinate and moves the shift inside the tanh, which turns a global squashing function into a dimension-wise adaptive gate. Our experiments include the original shared-α form as an explicit baseline.

Nested representations and evaluation. Matryoshka Representation Learning trains prefixes of one embedding to be independently usable, letting a single model serve several storage budgets [27]; both the reference baseline and SSE train with an MRL objective. BEIR is the standard zero-shot retrieval suite [28], and NanoBEIR is its compact 13-task subsample with roughly 50 queries per task [29], cheap enough to run during training and now the default benchmark for static-embedding releases.

3. SSE TRAINING RECIPE

3.1. Recipe Overview

SSE is a training recipe for a lookup-and-pool sentence encoder. Its model consists of a tokenizer, a learned token table, exact mean pooling, and a coordinate-wise post-pooling map. Training combines contrastive ranking, Matryoshka supervision at 32, 64, 128, 256, and 512 dimensions, a 15-source mixture, a large effective batch, and a one-epoch schedule. Algorithm 1 summarizes the forward pass, Figure 1 shows the architecture, and Section 4.1 specifies the optimization recipe.

The released setting uses Separable DyT (Dynamic Tanh normalization), a lightweight post-pooling transform derived from DyT [26]. It operates independently on each coordinate and adds negligible parameters. Appendix A evaluates this choice against six alternatives under paired controls.

The reference recipe of Aarsen [6] provides the lookup-and-mean-pooling foundation, the BERT tokenizer, contrastive ranking, and Matryoshka supervision. SSE changes the coupled recipe choices rather than a single isolated component: it uses a 512-dimensional table with a coordinate-wise post-pooling map, a registered 15-source mixture, an effective batch of 8,192, and a lower learning rate with cosine decay. Table 1 gives the side-by-side comparison together with the released training settings.

Received date: Aug. 30 2026
Accepted date: Sept. 15 2026

DOI: 10.63211/j.p.26.204501
ISSN 2760-8646 (Online) ICAITD

Input: $x : (B, S)$
Output: $y : (B, E)$
$x_T : (B, S) \leftarrow \text{Tokenizer}(x)$
$x_S : (B, E) \leftarrow \text{EmbeddingBag}(x_T)$
$y : (B, E) \leftarrow \text{SeparableDyT}(x_S)$
return y

Algorithm 1. Forward pass used by the released SSE recipe. B is the batch size, S the token-sequence length, and $E = d$ the embedding width.

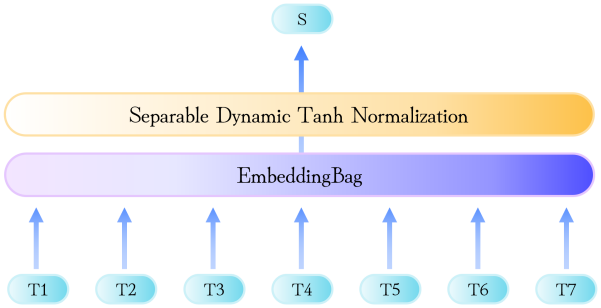


Fig.1. Architecture used by the released SSE recipe. Token representations are mean-pooled by `EmbeddingBag`, then mapped coordinate-wise to the sentence representation. The post-pooling map introduces no token-to-token interaction and adds $3d$ parameters.

3.2. Post-Pooling Transform

For each embedding dimension x_k, Separable DyT computes the output as

$$y_k = \gamma_k \tanh(\alpha_k x_k + \beta_k), \qquad k = 1, \ldots, d, \tag{1}$$

where α_k, β_k, and γ_k are learnable parameters that control scaling, shifting, and output amplitude respectively (Algorithm 2). In the released models they are initialised to 0.5, 0, and 1.

Input: $x : (B, E)$
Output: $y : (B, E)$
Parameters: $\alpha, \beta, \gamma : (E)$
$x \leftarrow \alpha \cdot x + \beta$
$y : (B, E) \leftarrow \gamma \cdot \text{Tanh}(x)$
return y

Algorithm 2. Separable DyT (Dynamic Tanh normalization). All three parameter vectors are per-coordinate, and the shift is applied *inside* the tanh.

The derivative with respect to the input dimension x_k is

$$\frac{\partial y_k}{\partial x_k} = \gamma_k \alpha_k \operatorname{sech}^2(\alpha_k x_k + \beta_k). \tag{2}$$

Let $z_k = \alpha_k x_k + \beta_k$. The derivative provides magnitude-dependent gradient gating through the squared hyperbolic secant:

- For saturated coordinates ($|z_k| \gg 1$), $\operatorname{sech}^2(z_k) \sim 4e^{-2|z_k|}$ and $\partial y_k / \partial x_k \to 0$.
- Near the linear region ($|z_k| \ll 1$), $\operatorname{sech}^2(z_k) \approx 1$ and $\partial y_k / \partial x_k \approx \gamma_k \alpha_k$.

Thus, the transform attenuates gradients when a coordinate enters the saturated region and preserves them near its linear region. Whether this behavior improves retrieval is tested empirically rather than assumed from the derivative.

3.3. Optimization Behavior

Unlike layer normalization or L_2 normalization, the post-pooling map does not globally rescale each embedding. Its learned per-coordinate parameters instead change the local derivative through $\gamma_k \alpha_k \operatorname{sech}^2(z_k)$. This property can alter optimization and allow a lower-rank structure to emerge during training. Section 5.2 examines the learned spectrum, while Appendix A reports component-level controls.

3.4. Integration into Static Embedding Models

Separable DyT is applied to the output of an `EmbeddingBag` layer, which aggregates token embeddings into a fixed-dimensional representation. Let $\mathbf{E} \in \mathbb{R}^{V \times d}$ denote the embedding matrix and let a sentence be represented by a set of token indices $\{t_1, \ldots, t_n\}$. The `EmbeddingBag` layer computes a pooled representation

$$\mathbf{z} = \text{EmbeddingBag}(t_1, \ldots, t_n) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{E}_{t_i}, \tag{3}$$

where the aggregation uses mean pooling over the selected embeddings. Separable DyT is then applied to the aggregated representation,

$$\mathbf{s} = \text{SeparableDyT}(\mathbf{z}), \tag{4}$$

where $\mathbf{s} \in \mathbb{R}^d$ denotes the final sentence embedding, scored with cosine similarity.

Because the post-pooling map operates element-wise on the aggregated vector, it preserves the encoder's simple static structure. The released model uses the `bert-base-uncased` tokenizer ($V = 30{,}522$) at $d = 512$, for 15.6M parameters in total; the map contributes $3d = 1{,}536$ parameters, under 0.01% of the model.

3.5. Computational Efficiency

SSE preserves the primary advantage of static embedding models: extremely fast inference. The `EmbeddingBag` operation performs efficient aggregation of token embeddings, avoiding the need for sequential computation or deep contextual encoding.

The Separable DyT transformation consists of element-wise affine transformations followed by a tanh activation, both of which are computationally inexpensive and highly parallelizable. Its cost is $O(d)$ per sentence against the $O(nd)$ cost of lookup and pooling, and the number of additional parameters grows linearly with the embedding dimension, remaining negligible compared to contextual encoders.

Consequently, the complete SSE model retains the inference structure of a static encoder. Section 5.3 reports its re-measured throughput.

Received date: Aug. 30 2026
Accepted date: Sept. 15 2026

DOI: 10.63211/j.p.26.204501
ISSN 2760-8646 (Online) ICAITD

4. EXPERIMENTS

4.1. Training Configuration

We train our embedding models using contrastive learning combined with Matryoshka Loss to support variable-dimensional embeddings. Following the training protocol of *Train 400x faster Static Embedding Models with Sentence Transformers* [6], we optimize for multiple projection dimensions simultaneously: 32, 64, 128, 256, and 512. This allows the model to maintain retrieval performance across different embedding sizes without retraining.

We use AdamW with BF16 mixed-precision training. The learning rate is 0.1 with cosine decay and a warmup ratio of 0.1. The released model uses a per-device batch size of 2,048 with 4 gradient accumulation steps, giving an effective batch size of 8,192. We train for exactly one epoch and record loss and gradient norm every 50 optimizer steps. To prevent duplicate pairs within a contrastive batch, we apply a `no_duplicates` sampler. The training objective is MultipleNegativesRankingLoss with in-batch negatives and cosine scoring, wrapped in a Matryoshka loss over the prefix dimensions above [27]. Table 1 combines the reference-recipe comparison with the released training settings. Appendix A gives the matched settings used in the 21-run component experiment.

Table 1. Released training recipe and its differences from the reference static-retrieval recipe of Aarsen [6].

Component	Reference recipe [6]	SSE released recipe
Tokenizer	BERT uncased	BERT uncased
Token table	Random initialization	Random initialization
Pooling	Mean	Mean
Objective	MNRL	MNRL
Embedding width	1024	512
Post-pooling map	None	Separable DyT ($3d$ parameters)
Training sources	13 English retrieval sources	Registered 15-source mixture
Source sampling	Proportional	Registered target ratios
Matryoshka widths	32, 64, 128, 256, 512, 768, 1024	32, 64, 128, 256, 512
Effective batch	2,048	8,192
Learning rate	0.2	0.1 with cosine decay
Training epochs	1	1
Precision	BF16	BF16
Batch sampler	`no_duplicates`	`no_duplicates`

4.2. Training Datasets

We train on 15 sources covering question answering (QA), natural language inference (NLI), and information retrieval (IR). Table 2 reports the registered target composition of the training mixture. Batches contain no duplicate queries and draw from one source at a time, with sources sampled according to these target ratios.

Table 2. Registered target composition of the 15-source training mixture.

Dataset	Domain	Ratio
squad	Question answering	2%
trivia_qa	Question answering	2%
allnli	Natural language inference	2%
pubmedqa	Scientific QA	2%
hotpotqa	Multi-hop QA	2%
miracl	Multilingual IR	2%
mr_tydi	Multilingual IR	2%
msmarco	Web search IR	5%
msmarco_10m	Large-scale IR	45%
msmarco_hard	Hard-negative IR	2%
mldr	Long-document retrieval	2%
s2orc	Scientific text	14%
swim_ir	Semantic web IR	2%
paq	Question answering	14%
nq	Natural Questions	2%

5. EVALUATION

We evaluate the complete SSE recipe in terms of dimensional and parameter efficiency, released-model retrieval quality, learned representation structure and its rank profile, and its position on the speed–quality frontier. We also report a BM25 lexical baseline on the same NanoBEIR tasks. The 21-run component experiment is reported in Appendix A.

5.1. Dimensional and Parameter Efficiency

The primary comparison is the quality retained at a smaller representation and model size. At 256 dimensions, SSE reaches 0.5036 and matches the prior static reference's native 1024-dimensional score of 0.5030 (Table 3). Each stored vector therefore uses one quarter as many values, and dense similarity scoring uses one quarter as many coordinate operations. The released SSE checkpoint contains 15.6M parameters, approximately half the reference's 31.3M. At its full 512-dimensional width, SSE reaches 0.5159; the corresponding +0.0129 native-reference difference is secondary to the dimensional and parameter reduction.

Table 3. NanoBEIR mean nDCG@10 by retained dimension for single-run released checkpoints measured with the same evaluator. The 256-dimensional SSE prefix matches the native 1024-dimensional reference (0.5036 vs. 0.5030) with one quarter of the vector storage; the released checkpoints contain 15.6M and 31.3M parameters, respectively.

System	32	64	128	256	512	1024
SSE (released recipe)	0.3487	0.4236	0.4732	0.5036	0.5159	–
static-retrieval-mrl-en-v1	0.3532	0.4177	0.4621	0.4818	0.4958	0.5030

5.2. Representation Improvement Beyond Rank Reduction

The released SSE checkpoint develops a spectral cutoff near component 430 during training without an explicit rank constraint. The near-full-rank `sepdyt_no_prune` variant nevertheless matches `sepdyt` at 0.5097, while removing the learned map from the trained SSE checkpoint reduces nDCG@10 by 0.0098. By contrast, forcing rank 432 yields 0.5068 and does not reproduce the gain (Appendix A.2). Together, these controls show that the learned post-pooling map reorganizes the representation to improve retrieval. Lower-rank structure can emerge as a result of this learned

Received date: Aug. 30 2026
Accepted date: Sept. 15 2026

DOI: 10.63211/j.p.26.204501
ISSN 2760-8646 (Online) ICAITD

improvement, but imposing the same rank structure does not create it.

5.3. Speed–Quality Pareto Frontier

Table 4 and Figure 2 report the speed comparison, which we re-measured for this paper instead of using single timed runs. We measure the released SSE checkpoint under the benchmark conditions of the reference static-embeddings release: GooAQ query embeddings per second on CPU [6]. We extend that protocol in three ways. First, each system uses its own throughput-optimal batch size, selected by a per-model sweep. This avoids disadvantaging a system with a batch size selected for another model. Static models saturate at batch sizes of 2048–8192, whereas all transformers peak at 128 and become 1.3–1.9× slower at 2048. Second, we repeat every measurement to obtain a confidence interval. Third, we measure the static systems in interleaved rounds with rotating order so that changes in machine load affect every system equally.

The comparison includes a broad set of static models rather than a single representative. It includes both Sentence Transformers reference releases [4, 5]; Model2Vec distillations from `potion-base-2M` to `potion-base-32M`; retrieval-tuned and multilingual variants [8–10]; and the GloVe distillation `M2V_base_glove` [11]. An untrained mean-pooled GloVe table [12] provides a baseline before contrastive training. Across these eleven systems, throughput ranges from 110,084 to 133,205 queries/s, a factor of 1.21. In contrast, their parameter counts differ by more than an order of magnitude (1.9M to 128M), and their nDCG@10 scores differ by 0.19. Thus, within this family, throughput is almost independent of quality.

These results place SSE on the measured dense-encoder speed–quality Pareto frontier (Table 4, Figure 2). No measured dense embedding encoder is simultaneously faster and more accurate. SSE encodes 119,832 queries/s and has the highest measured quality among the throughput-comparable static systems, while every higher-scoring contextual model is 45–286× slower. Throughput intervals within the static family overlap, but the measured points preserve this frontier relation.

Received date: Aug. 30 2026
Accepted date: Sept. 15 2026

DOI: 10.63211/j.p.26.204501
ISSN 2760-8646 (Online) ICAITD

Table 4. Re-measured CPU encoding throughput (queries per second, mean ± Student-t 95% interval) and NanoBEIR nDCG@10 for dense embedding models, following the benchmark conditions of the reference static-embeddings release [6]: GooAQ query embeddings per second on an Intel Core Ultra 7 265K. "Dim." and "Params" are the native output width and checkpoint parameter count. Each system uses its throughput-optimal batch size. Static systems are measured in seven interleaved rounds over 100,000 queries and transformers in three repetitions over 10,000 queries. "× ref." is the throughput ratio to `static-retrieval-mrl-en-v1`; all nDCG@10 values use one evaluator. BM25 is excluded because it scores a sparse inverted index rather than encoding dense vectors.

Model	Dim.	Params	nDCG@10	Batch	Queries/s	× ref.
SSE (English)	512	15.6M	0.5159	8192	119,832 ± 7,302	1.05
potion-retrieval-32M	512	32.3M	0.5108	2048	120,271 ± 7,872	1.05
static-retrieval-mrl-en-v1	1024	31.3M	0.5030	4096	114,223 ± 7,326	1.00
potion-base-32M	512	32.3M	0.4640	1024	118,318 ± 11,473	1.04
potion-base-8M	256	7.6M	0.4427	2048	118,879 ± 2,128	1.04
static-similarity-mrl-multilingual-v1	1024	108.4M	0.4350	8192	110,084 ± 3,574	0.96
potion-base-4M	128	3.8M	0.4230	2048	133,205 ± 10,825	1.17
M2V_base_glove	256	102.4M	0.4077	4096	132,093 ± 12,810	1.16
potion-multilingual-128M	256	128.1M	0.3701	8192	111,386 ± 6,832	0.98
potion-base-2M	64	1.9M	0.3668	1024	128,900 ± 9,131	1.13
glove.6B.300d (mean pool)	300	120.0M	0.3300	1024	111,360 ± 7,076	0.97
all-MiniLM-L6-v2	384	22.7M	0.5624	128	2,689 ± 278	0.024
multilingual-e5-small	384	117.7M	0.5546	128	1,380 ± 263	0.012
bge-base-en-v1.5	768	109.5M	0.6322	128	419 ± 6	0.004

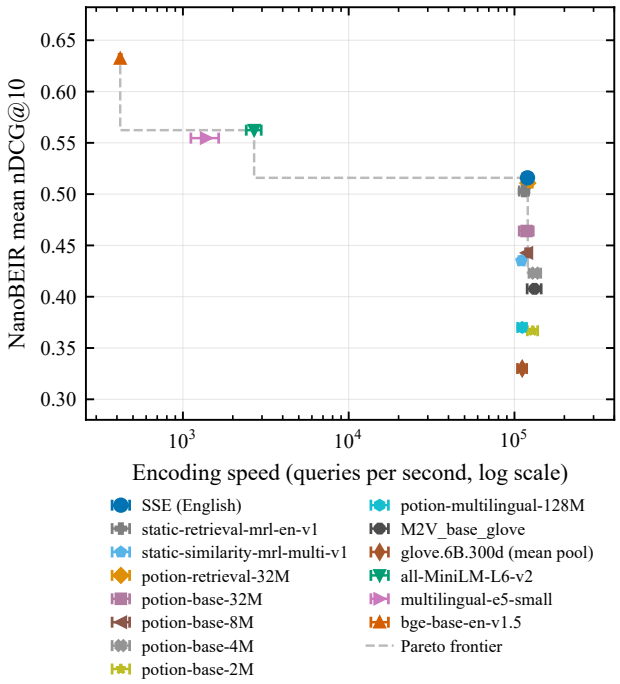


Fig.2. NanoBEIR quality against re-measured CPU encoding speed for dense embedding models (GooAQ queries, per-model batch size; log scale; horizontal bars are throughput 95% intervals). Throughput within the static family varies by a factor of 1.21 and intervals overlap. No measured dense encoder is simultaneously faster and more accurate than SSE, placing it on the measured speed–quality Pareto frontier; every higher-scoring contextual system encodes 45–286× slower.

5.4. Dense and Lexical Per-Task Results

Table 1 shows that both recipes use the BERT uncased tokenizer, a randomly initialized token table, mean pooling, MNRL, Matryoshka supervision, one epoch, BF16, and no-duplicate batches, while isolating the choices changed by SSE. The contribution is the combined training recipe; the Appendix isolates only the post-pooling component.

The released 512-dimensional SSE model exceeds the native 1024-dimensional dense reference on 8 of 13 tasks (Table 5). For a lexical baseline, we evaluate BM25 with lowercase Unicode word tokenization and standard $k_1 = 1.2$, $b = 0.75$ settings [13], using the same NanoBEIR corpora, queries, relevance judgments, and nDCG@10 definition. BM25 scores 0.5457 on average and exceeds SSE on 9 of 13 tasks, while SSE is highest on ArguAna, NQ, QuoraRetrieval, and SCIDOCS. This result shows that the parameter- and storage-efficiency claim concerns dense retrieval; lexical matching remains a strong complementary baseline.

Table 5. Per-task NanoBEIR English nDCG@10 for the released SSE checkpoint at 512 dimensions, the native 1024-dimensional dense reference, and BM25. BM25 uses lowercase Unicode word tokens, $k_1 = 1.2$, and $b = 0.75$ on the same corpora, queries, and relevance judgments. The highest score in each row is bold.

Task	SSE 512	Reference 1024	BM25
ArguAna	**0.4096**	0.4064	0.3482
ClimateFEVER	0.2941	**0.3309**	0.3139
DBPedia	0.5503	**0.5681**	0.5588
FEVER	0.6810	0.6922	**0.8176**
FiQA2018	0.3499	**0.3651**	0.3605
HotpotQA	0.7105	0.6547	**0.8387**
MSMARCO	0.4162	0.4041	**0.4825**
NFCorpus	0.3145	0.3242	**0.3289**
NQ	**0.4790**	0.4534	0.4718
QuoraRetrieval	**0.9171**	0.8949	0.7893
SCIDOCS	**0.3548**	0.2643	0.3420
SciFact	0.6493	0.6111	**0.7178**
Touche2020	0.5804	0.5703	**0.7240**
Mean	0.5159	0.5030	**0.5457**

6. DISCUSSION

6.1. Future Work

Future work should evaluate and separate the recipe's contributions at higher resolution:

- **Higher-resolution retrieval evaluation.** Full BEIR, more paired seeds, and query- or task-level bootstrap intervals would better resolve differences below the NanoBEIR noise scale.
- **Recipe attribution.** Factorial studies should separate the effects of the data mixture, Matryoshka

Received date: Aug. 30 2026
Accepted date: Sept. 15 2026

DOI: 10.63211/j.p.26.204501
ISSN 2760-8646 (Online) ICAITD

supervision, effective batch, schedule, optimizer, and post-pooling choice, including fixed post-processing alternatives such as ABTT and whitening.

- **Deployment measurements.** Future benchmarks should add single-query and p99 latency, thread and BLAS settings, index memory, and quantized quality–storage curves.

7. LIMITATIONS

The evidence is limited to English and the compact 13-task NanoBEIR suite (about 650 queries). Training and evaluation cover similar domains and task families: MS MARCO, NQ, and HotpotQA contribute training data and have corresponding NanoBEIR tasks, while scientific-text sources align with NanoBEIR scientific-retrieval tasks. However, as in the reference recipe, the NanoBEIR evaluation examples themselves are excluded from training [6]. No evaluation example is therefore directly reused, although the domain similarity means that NanoBEIR is not a fully out-of-domain evaluation.

Complete-recipe quality comparisons are single-run point estimates; the three-seed component study has limited resolution for small effects. Throughput is measured on one CPU host, and its intervals overlap within the static family. BM25 uses one stated tokenizer and untuned standard parameters; sparse-index size and end-to-end lexical latency are not compared with dense retrieval. Results therefore apply to this benchmark and system setting; transfer across domains, hardware, and languages remains to be established.

8. CONCLUSION

We presented SSE, a parameter-efficient training recipe for static dense retrieval. Its 256-dimensional prefix matches the native 1024-dimensional dense reference (0.5036 vs. 0.5030) with one quarter of the vector storage, while the 15.6M-parameter checkpoint uses approximately half the reference's 31.3M parameters. The full 512-dimensional representation reaches 0.5159, and the released model lies on the measured dense-encoder CPU speed–quality Pareto frontier. Component controls further show that the learned post-pooling map improves the representation, while the observed lower-rank spectrum emerges from this learning rather than causing the gain.

Acknowledgements

The author acknowledges the support of Saldra, Witness, and Lumina Logic Minds for providing the computational resources used in this work. Our interest in this topic originated from reading Tom Aarsen's article *Train 400x faster Static Embedding Models with Sentence Transformers* [6], which motivated us to investigate static embeddings. We thank the developers of `sentence-transformers`, Python, and PyTorch, the maintainers of the BEIR and NanoBEIR datasets and of the Model2Vec and `potion` releases, and all the researchers whose efforts this work builds upon.

Impact Statement

The main impact of SSE is computational efficiency. Compact static embeddings can reduce retrieval latency, vector storage, model memory, and inference energy, enabling search on local and resource-constrained systems. The 256-dimensional representation uses one quarter of the vector storage of a 1024-dimensional reference at comparable NanoBEIR quality, and the released checkpoint uses approximately half as many parameters. However, the registered training mixture is English-dominant and assigns 52% to the MS MARCO family, so quality and bias behavior should not be assumed to transfer to other languages, domains, or high-stakes decisions. Indexing private corpora also requires appropriate access control and data governance. This work collects no new personal data and does not evaluate uses that directly determine outcomes for individuals.

References

[1] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013. [Online]. Available: https://arxiv.org/abs/1301.3781

[2] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 2014, pp. 1532–1543.

[3] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.

[4] Sentence Transformers, "static-retrieval-mrl-en-v1," Hugging Face model card, 2025, accessed July 28, 2026. [Online]. Available: https://huggingface.co/sentence-transformers/static-retrieval-mrl-en-v1

[5] ——, "static-similarity-mrl-multilingual-v1," Hugging Face model card, 2025, accessed August 26, 2026. [Online]. Available: https://huggingface.co/sentence-transformers/static-similarity-mrl-multilingual-v1

[6] T. Aarsen, "Train 400x faster static embedding models with sentence transformers," Hugging Face Blog, Jan. 2025, published January 15, 2025. [Online]. Available: https://huggingface.co/blog/static-embeddings

[7] S. Arora, Y. Liang, and T. Ma, "A simple but tough-to-beat baseline for sentence embeddings," in *International Conference on Learning Representations*, 2017. [Online]. Available: https://openreview.net/forum?id=SyK00v5xx

[8] MinishLab, "Model2Vec: Distill a small fast model from any sentence transformer," Software library and model releases (`potion` series), 2024, accessed August 26, 2026. [Online]. Available: https://github.com/MinishLab/model2vec

[9] ——, "potion-base static embedding models (2M, 4M, 8M, 32M)," Hugging Face model cards, 2024,

Received date: Aug. 30 2026
Accepted date: Sept. 15 2026

DOI: 10.63211/j.p.26.204501
ISSN 2760-8646 (Online) ICAITD

accessed August 29, 2026. [Online]. Available: https://huggingface.co/collections/minishlab/potion

[10] ——, "potion-multilingual-128m," Hugging Face model card, 2025, accessed August 29, 2026. [Online]. Available: https://huggingface.co/minishlab/potion-multilingual-128M

[11] ——, "M2V_base_glove: a Model2Vec distillation of GloVe," Hugging Face model card, 2024, accessed August 29, 2026. [Online]. Available: https://huggingface.co/minishlab/M2V_base_glove

[12] Sentence Transformers, "average_word_embeddings_glove.6b.300d," Hugging Face model card, 2019, accessed August 29, 2026. [Online]. Available: https://huggingface.co/sentence-transformers/average_word_embeddings_glove.6B.300d

[13] S. Robertson and H. Zaragoza, "The probabilistic relevance framework: BM25 and beyond," *Foundations and Trends in Information Retrieval*, vol. 3, no. 4, pp. 333–389, 2009.

[14] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using siamese BERT-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, 2019, pp. 3982–3992.

[15] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou, "MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers," in *Advances in Neural Information Processing Systems*, vol. 33, 2020. [Online]. Available: https://arxiv.org/abs/2002.10957

[16] L. Wang, N. Yang, X. Huang, B. Jiao, L. Yang, D. Jiang, R. Majumder, and F. Wei, "Text embeddings by weakly-supervised contrastive pre-training," *arXiv preprint arXiv:2212.03533*, 2022. [Online]. Available: https://arxiv.org/abs/2212.03533

[17] L. Wang, N. Yang, X. Huang, L. Yang, R. Majumder, and F. Wei, "Multilingual E5 text embeddings: A technical report," *arXiv preprint arXiv:2402.05672*, 2024. [Online]. Available: https://arxiv.org/abs/2402.05672

[18] S. Xiao, Z. Liu, P. Zhang, and N. Muennighoff, "C-Pack: Packaged resources to advance general Chinese embedding," in *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2024. [Online]. Available: https://arxiv.org/abs/2309.07597

[19] S. Hofstätter, S. Althammer, M. Schröder, M. Sertkan, and A. Hanbury, "Improving efficient neural ranking models with cross-architecture knowledge distillation," *arXiv preprint arXiv:2010.02666*, 2020. [Online]. Available: https://arxiv.org/abs/2010.02666

[20] T. Wang and P. Isola, "Understanding contrastive representation learning through alignment and uniformity on the hypersphere," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 119, 2020, pp. 9929–9939. [Online]. Available: https://proceedings.mlr.press/v119/wang20k.html

[21] K. Ethayarajh, "How contextual are contextualized word representations? comparing the geometry of BERT, ELMo, and GPT-2 embeddings," *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pp. 55–65, 2019.

[22] J. Mu, S. Bhat, and P. Viswanath, "All-but-the-top: Simple and effective postprocessing for word representations," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=HkuGJ3kCb

[23] B. Li, H. Zhou, J. He, M. Wang, Y. Yang, and L. Li, "On the sentence embeddings from pre-trained language models," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, 2020, pp. 9119–9130.

[24] J. Su, J. Cao, W. Liu, and Y. Ou, "Whitening sentence representations for better semantics and faster retrieval," *arXiv preprint arXiv:2103.15316*, 2021. [Online]. Available: https://arxiv.org/abs/2103.15316

[25] T. Gao, X. Yao, and D. Chen, "SimCSE: Simple contrastive learning of sentence embeddings," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021, pp. 6894–6910.

[26] J. Zhu, X. Chen, K. He, Y. LeCun, and Z. Liu, "Transformers without normalization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025. [Online]. Available: https://arxiv.org/abs/2503.10622

[27] A. Kusupati, G. Bhatt, A. Rege, M. Wallingford, A. Sinha, V. Ramanujan, W. Howard-Snyder, K. Chen, S. Kakade, P. Jain, and A. Farhadi, "Matryoshka representation learning," in *Advances in Neural Information Processing Systems*, vol. 35, 2022.

[28] N. Thakur, N. Reimers, A. Rücklé, A. Srivastava, and I. Gurevych, "BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models," in *Thirty-fifth Conference on Neural Information Processing Systems, Datasets and Benchmarks Track*, 2021. [Online]. Available: https://openreview.net/forum?id=wCu6T5xFjeJ

[29] Sentence Transformers, "NanoBEIR evaluator documentation," Sentence Transformers documentation, 2026, accessed July 28, 2026. [Online]. Available: https://sbert.net/docs/package_reference/sentence_transformer/evaluation.html

Received date: Aug. 30 2026
Accepted date: Sept. 15 2026

DOI: 10.63211/j.p.26.204501
ISSN 2760-8646 (Online) ICAITD

A. PAIRED COMPONENT EXPERIMENT

We quantify post-pooling design choices within the SSE recipe using a paired three-seed experiment (42, 44, 123) at 512 dimensions. Within each seed, all seven configurations share the byte-identical initial embedding table (SHA-256 verified), the same 10,884,622 training pairs in the same order, the data manifest, and all optimization settings. The component runs use a per-device batch size of 512 with 8 accumulation steps (effective batch 4,096). Configuration hashes were verified for every run, and all 21 runs completed. The configurations and six contrasts of Appendix A.2 were fixed before results were inspected.

Configurations. Table 6 compares the no-operator baseline, shared DyT, Separable DyT, and four diagnostic variants that isolate the affine map, factorization, coordinate silencing, and fixed rank.

Table 6. Seven coordinate-wise post-pooling configurations on $z \in \mathbb{R}^{512}$. Learned gains start at 0.5 or 1 and biases at 0.

Configuration	Operator	Learned	Purpose
`none`	$y = z$	—	recipe alone
`shared_dyt`	$y = w \odot \tanh(\alpha z) + b$	$\alpha \in \mathbb{R}$; $w, b \in \mathbb{R}^d$	DyT as proposed [26]
`diag_affine`	$y = g \odot z + b$	g, b	centering + gain, no tanh
`factor_affine`	$y = \gamma \odot (\alpha \odot z + b)$	α, γ, b	Separable DyT minus tanh
`sepdyt`	$y = \gamma \odot \tanh(\alpha \odot z + b)$	α, γ, b	released post-pooling map
`sepdyt_no_prune`	`sepdyt` with $\lvert\alpha_k\rvert, \lvert\gamma_k\rvert \geq \sqrt{10^{-3}}$	α, γ, b	forbid dead coordinates
`fixed_tail_rank`	$y = m \odot z$, m fixed, $m_k = \mathbf{1}[k \leq 432]$	—	impose low rank directly

The shared-DyT baseline uses the original Dynamic Tanh parameterization in Algorithm 3.

Input: $x : (B, E)$
Output: $y : (B, E)$
Parameters: $\alpha : (1)$; $\beta, \gamma : (E)$
1: $x \leftarrow \mathrm{Tanh}(\alpha \cdot x)$
2: $y : (B, E) \leftarrow \gamma \cdot x + \beta$
3: **return** y

Algorithm 3. Original Dynamic Tanh normalization [26], used as the shared-α baseline.

Evaluation. All configurations are evaluated on the same fixed NanoBEIR-en revision. The benchmark contains 13 English tasks with about 50 queries each. Scores use cosine similarity over L_2-normalised vectors and binary nDCG@10 at $k = 10$, averaged equally across tasks. We report three-seed means and Student-t 95% intervals; seed-level differences are ordered 42 / 44 / 123. Numerical rank counts covariance eigenvalues above 10^{-6} of the maximum on a task-balanced probe, and silent coordinates satisfy $|\gamma_k \alpha_k| < 10^{-3}$.

A.1. Training Dynamics

The `sepdyt` configuration continues improving during the final quarter of training, matches no operator around step 2,250, and finishes at 0.5097, compared with 0.5085 for no operator and 0.5088 for shared DyT (Figure 3). The final differences remain within seed variation.

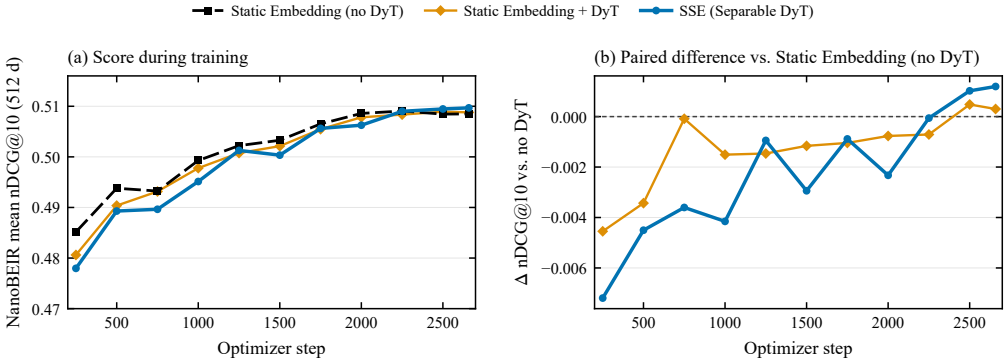


Fig.3. **(a)** NanoBEIR mean nDCG@10 at 512 dimensions across training (three-seed means, recorded every 250 optimizer steps). **(b)** Paired differences against no operator.

Received date: Aug. 30 2026
Accepted date: Sept. 15 2026

DOI: 10.63211/j.p.26.204501
ISSN 2760-8646 (Online) ICAITD

A.2. Component-Level Results

Overall comparison. At 512 dimensions, `sepdyt` and `sepdyt_no_prune` share the highest mean (0.5097); Table 7 gives all seven configurations.

Table 7. Extended comparison at 512 dimensions across the seven configurations. "Rank" is numerical rank at relative eigenvalue threshold 10^{-6}; "Silent" counts coordinates with $|\gamma_k \alpha_k| < 10^{-3}$.

Configuration	Mean	95% CI	Rank	Silent
`sepdyt`	**0.5097**	[.4947, .5247]	443.3	67.3
`sepdyt_no_prune`	**0.5097**	[.4949, .5246]	510.7	0.0
`shared_dyt`	0.5088	[.4966, .5210]	479.0	32.0
`diag_affine`	0.5086	[.4915, .5257]	445.3	66.7
`none`	0.5085	[.4958, .5212]	512.0	0.0
`factor_affine`	0.5076	[.4951, .5202]	442.7	58.3
`fixed_tail_rank`	0.5068	[.4987, .5150]	432.0	80.0

Task-level results. The aggregate differences are not uniform across tasks. At 512 dimensions, `sepdyt` exceeds no operator on 6 of 13 tasks and shared DyT on 8 of 13 tasks (Table 8).

Table 8. Per-task NanoBEIR English nDCG@10 at 512 dimensions (three-seed means) for the released post-pooling configuration and its two principal baselines.

Configuration	ArguAna	Climate FEVER	DBPedia	FEVER	FiQA 2018	Hotpot QA	MS MARCO	NF Corpus	NQ	Quora Retrieval	SCI DOCS	SciFact	Touche 2020	Mean
`sepdyt`	0.3815	0.3043	0.5569	0.6846	0.3497	0.6957	0.4352	0.3036	0.4506	0.9011	0.3413	0.6250	0.5965	**0.5097**
`shared_dyt`	0.3890	0.3012	0.5549	0.6840	0.3534	0.6968	0.4395	0.3025	0.4483	0.9052	0.3395	0.6104	0.5894	0.5088
`none`	0.3890	0.3028	0.5524	0.6866	0.3528	0.7023	0.4400	0.3010	0.4542	0.9020	0.3395	0.5985	0.5893	0.5085

Retained dimensions. The three principal configurations remain close across the Matryoshka prefixes (Table 9). Their ordering changes below 512 dimensions, so the small final-width difference should not be treated as a uniform advantage across storage budgets.

Table 9. Controlled NanoBEIR mean nDCG@10 by retained dimension (three-seed means; best result at each dimension in bold).

Configuration	32	64	128	256	512
`sepdyt`	0.3410	**0.4162**	0.4612	0.4925	**0.5097**
`shared_dyt`	**0.3419**	0.4138	0.4632	0.4943	0.5088
`none`	0.3393	0.4121	**0.4649**	**0.4962**	0.5085

Paired contrasts. The six contrasts were fixed before results were inspected. Table 10 reports paired means, Student-t 95% intervals, and seed-level differences; with three seeds, the exact sign-flip floor is $p = 0.25$.

Table 10. Preregistered contrasts, NanoBEIR, 512 dimensions, three paired seeds.

Contrast	Δ	95% CI	Per-seed Δ (42 / 44 / 123)
`sepdyt − none`	+0.0012	[−0.0026, +0.0050]	+0.0024 / −0.0005 / +0.0018
`sepdyt − factor_affine`	+0.0020	[−0.0032, +0.0073]	+0.0035 / −0.0004 / +0.0030
`sepdyt − shared_dyt`	+0.0009	[−0.0036, +0.0054]	+0.0023 / −0.0011 / +0.0015
`factor_affine − diag_affine`	−0.0010	[−0.0056, +0.0036]	−0.0030 / −0.0006 / +0.0006
`fixed_tail_rank − none`	−0.0017	[−0.0064, +0.0031]	−0.0037 / −0.0014 / +0.0001
`sepdyt − sepdyt_no_prune`	−0.00004	[−0.0003, +0.0002]	+0.0000 / +0.0000 / −0.0002

A.3. Emergent Rank from Representation Improvement

The released checkpoint develops a spectral cutoff near component 430 without an explicit rank constraint (Figure 4). However, `sepdyt` and `sepdyt_no_prune` share the highest mean of 0.5097 at numerical ranks 443.3 and 510.7, respectively. The learned improvement is therefore realized with distinct rank profiles: lower-rank structure is one outcome of representation learning, rather than the cause of the retrieval gain.

The operator-removal and fixed-rank controls locate the gain in the learned representation. Removing the learned post-pooling map from the trained Separable DyT checkpoint reduces mean nDCG@10 by 0.0098, whereas forcing rank

Received date: Aug. 30 2026
Accepted date: Sept. 15 2026

DOI: 10.63211/j.p.26.204501
ISSN 2760-8646 (Online) ICAITD

432 scores 0.5068 versus 0.5085 for no operator, a paired difference of −0.0017 with a 95% interval of [−0.0064, +0.0031]. A prescribed rank mask therefore cannot reproduce the co-adapted representation learned by the embedding table and post-pooling map; applying low-rank reduction after learning does not recover the observed improvement.

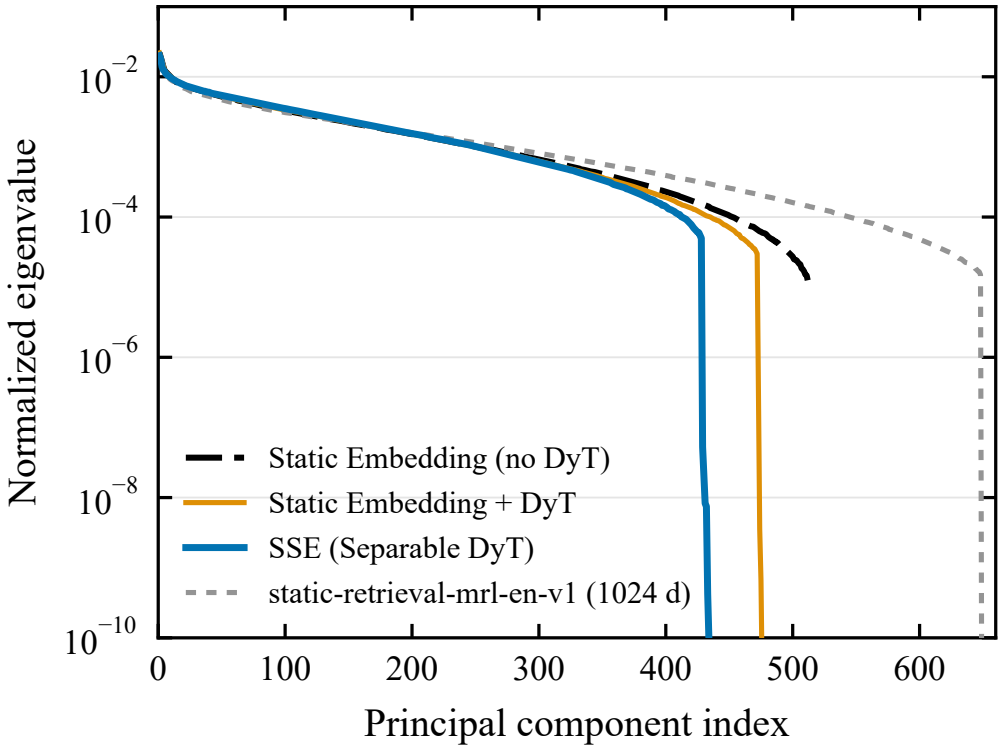


Fig.4. PCA spectra of NanoBEIR query embeddings from the released checkpoints, using a task-balanced sample across the 13 English tasks. The no-operator model decays smoothly and remains numerically full rank; shared DyT cuts off near component 473, and the Separable DyT model cuts off earlier, near component 430. The 1024-dimensional reference is included for scale.

Removing the learned post-pooling map from a trained checkpoint reduces mean nDCG@10 by 0.0098 for Separable DyT and by 0.0063 for shared DyT at 512 dimensions (Figure 5). This drop shows that the embedding table and learned map form a co-adapted representation; the map is part of the learned representation improvement rather than an interchangeable post-hoc rank operator.

Received date: Aug. 30 2026
Accepted date: Sept. 15 2026

DOI: 10.63211/j.p.26.204501
ISSN 2760-8646 (Online) ICAITD

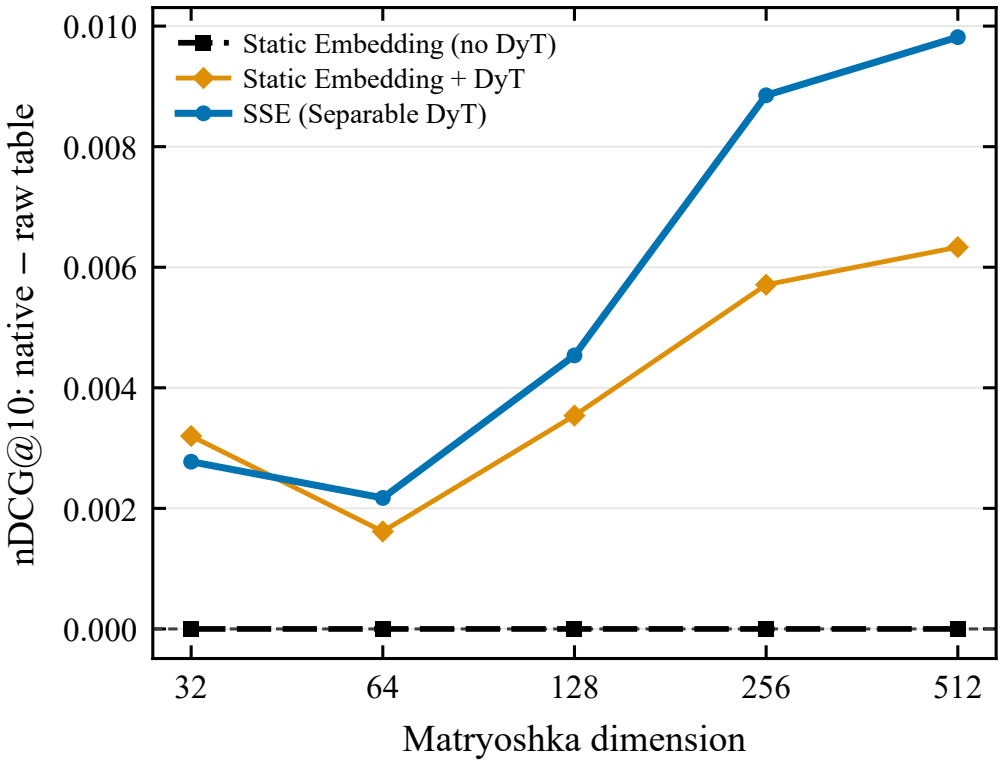


Fig.5. Within-checkpoint dependence on the post-pooling map: native-representation nDCG@10 minus raw-table nDCG@10 at each retained dimension (three-seed means). At 512 dimensions, the differences are +0.0098 for Separable DyT and +0.0063 for shared DyT; the no-operator configuration is identically zero.

A.4. Seed-Level Interpretation

At 512 dimensions, the paired means are 0.5097 for `sepdyt`, 0.5088 for shared DyT, and 0.5085 for no operator. The paired differences are +0.0009 and +0.0012, respectively. Across seeds 42/44/123, the differences are +0.0023/−0.0011/+0.0015 against shared DyT and +0.0024/−0.0005/+0.0018 against no operator (Table 10). `sepdyt` is higher in two of the three matched seeds in both comparisons, but the ordering reverses for seed 44. The 95% intervals, [−0.0036, +0.0054] and [−0.0026, +0.0050], include zero, and the point estimates are smaller than the seed standard deviations of 0.003–0.007. The matched means therefore favor `sepdyt`, but the difference lies within seed variation and does not establish a seed-independent advantage for this component alone.

Received date: Aug. 30 2026
Accepted date: Sept. 15 2026

DOI: 10.63211/j.p.26.204501
ISSN 2760-8646 (Online) ICAITD