Early Exit Strategies for Approximate k-NN Search in Dense Retrieval

Francesco Busolin*
Ca' Foscari University
Venice, Italy

Claudio Lucchese
Ca' Foscari University
Venice, Italy

Franco Maria Nardini
ISTI-CNR
Pisa, Italy

Salvatore Orlando
Ca' Foscari University
Venice, Italy

Raffaele Perego
ISTI-CNR
Pisa, Italy

Salvatore Trani
ISTI-CNR
Pisa, Italy

Abstract

Learned dense representations are a popular family of techniques for encoding queries and documents using high-dimensional embeddings, which enable retrieval by performing approximate k nearest-neighbors search (A-kNN). A popular technique for making A-kNN search efficient is based on a two-level index, where the embeddings of documents are clustered offline and, at query processing, a fixed number N of clusters closest to the query is visited exhaustively to compute the result set.

In this paper, we build upon state-of-the-art for early exit A-kNN and propose an unsupervised method based on the notion of *patience*, which can reach competitive effectiveness with large efficiency gains. Moreover, we discuss a cascade approach where we first identify queries that find their nearest neighbor within the closest $\tau \ll N$ clusters, and then we decide how many more to visit based on our patience approach or other state-of-the-art strategies. Reproducible experiments employing state-of-the-art dense retrieval models and publicly available resources show that our techniques improve the A-kNN efficiency with up to 5× speedups while achieving negligible effectiveness losses. All the code used is available at https://github.com/francescobusolin/faiss_pEE.

CCS Concepts

• **Information systems** → **Information retrieval**; **Retrieval effectiveness**; **Retrieval efficiency**;

Keywords

Neural IR, Dense Retrieval, Efficiency/Effectiveness Trade-offs

ACM Reference Format:
Francesco Busolin, Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, and Salvatore Trani. 2024. Early Exit Strategies for Approximate k-NN Search in Dense Retrieval. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management (CIKM '24), October 21–25, 2024, Boise, ID, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3627673.3679903

*Corresponding author: francesco.busolin@unive.it

CIKM '24, October 21–25, 2024, Boise, ID, USA

ACM ISBN 979-8-4007-0436-9/24/10
https://doi.org/10.1145/3627673.3679903

1 Introduction

The recent developments in pretrained large language models (PLM) have shown the effectiveness of learned representations for many tasks, including ad-hoc text retrieval. In this context, one common approach relies on learned "dense" representations, where neural encoders are used to independently compute query and document representations in the same latent vector space. So far, two different kinds of dense representation have emerged: *single*-vector representations, where queries and documents are encoded with a single embedding [12, 14, 16, 27, 34, 37, 39], and *multi*-vector representations, where, conversely, questions and documents are represented with multiple embeddings [11, 18, 28, 29].

We focus on state-of-the-art single-vector representations. Given a collection of pre-computed document embeddings, retrieval of relevant documents for a query embedding becomes finding the set of documents that maximizes a similarity score, e.g., inner product, or minimizes a distance, e.g., Euclidean distance. Top-k retrieval thus becomes the problem of finding the k closest objects to the query in the multidimensional latent vector space, i.e., the k nearest neighbors (k-NN) points.

Related Work. Approximate k nearest neighbors search techniques trade-off accuracy for efficiency by avoiding scanning the entire collection thus accepting some loss in result quality [8, 20, 22, 30]. Indexing data structures for efficient A-kNN search rely on quantization or hashing/binning techniques [1, 25]. They typically partition the data points into disjoint clusters and perform a two-step search. First, the N clusters whose centroids turn out to be closest to the query are identified, where N is a static A-kNN hyperparameter commonly called *number of probes*. Second, these N closest clusters are visited exhaustively to identify the k closest data points to the query vector. The computational cost of A-kNN is proportional to the number and cardinality of the clusters visited, while the approximation error of the results retrieved decreases by increasing the number of clusters visited.

To further reduce the computational cost of A-kNN, Li *et al.* [19] proposed an *adaptive* A-kNN technique that dynamically chooses the number of clusters to probe on a per-query basis. The rationale of this strategy is to reduce the average query latency by limiting the number of clusters visited for *easy queries*, compared to the common A-kNN strategy that always probes a fixed number N of clusters for all queries. The technique relies on a learned regression model to estimate the number of probes and may be classified as an early-exit method for A-kNN, as it aims to stop the inspection

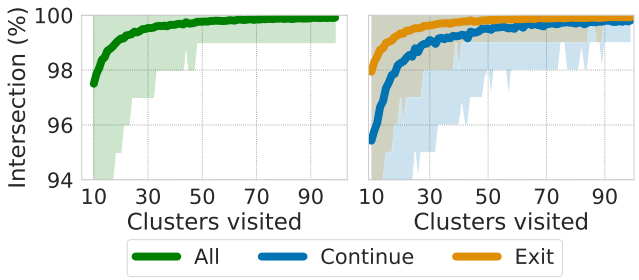

Figure 1: Average intersection size between consecutive result sets. The shaded area shows the region between the 5th and 95th percentile. The subdivision into *Exit* and *Continue* is obtained with $\tau = 10$.

of the clusters before reaching the N^{th} one. Other early-exit strategies for ad-hoc search have been investigated earlier [3–5, 35, 36]. Specifically, these strategies focus on document re-ranking with learning-to-rank models based on additive ensembles of regression trees and passage re-ranking models based on pre-trained language models. All these proposals aim to improve efficiency by selectively stopping the scoring process for documents that are likely not included among the top-k ones. In this paper, we further elaborate the method proposed by Li *et al.* [19] to introduce *adaptiveness* into the A-kNN algorithms by proposing novel early-exit methods for retrieval systems based on state-of-the-art single-representation dense models. Reproducible experiments conducted using three state-of-the-art dense representations, i.e., STAR [39], CONTRIEVER [14], and TAS-B [12] on the MS-MARCO dataset [24] show that our techniques improve the retrieval efficiency with *speedups* ranging from 4.71× to 5.27× over the FAISS-based [15] A-kNN baseline.

2 Problem Statement & Methodology

Problem Statement. Given a collection of embeddings $\mathcal{D} \subset \mathbb{R}^n$ and a query q embedded in the same latent space, we aim at identifying the k elements in $\mathcal{D}$ that are closest to q according to a similarity score $\sigma(q, d)$. In the case of dense retrieval models, the latent space is learned, and the function $\sigma(q, d)$ becomes a proxy of *query-document relevance*. Therefore, the set $k\text{NN}(q, \mathcal{D})$ computed using σ likely includes the k most relevant documents for q. However, computing the *exact* result set $k\text{NN}(q, \mathcal{D})$ can become computationally very expensive with large datasets composed of high dimensional vectors [13, 21, 33]. Thus, practical search systems commonly exploit approximate A-kNN algorithms, which rely on a precomputed partition of $\mathcal{D}$, and probe only the small subset of N clusters whose centroids are the most similar to the query.

Let $\mathcal{Q}$ be a set of queries and $q_i \in \mathcal{Q}$ be a generic query. We denote by d_i the document of $\mathcal{D}$ the most similar to q_i retrieved by an A-kNN algorithm for a fixed number of probes N, i.e., $d_i = A\text{-}1\text{NN}(q_i, \mathcal{D})$. We also denote by $d_i{}^*$ the actual document of $\mathcal{D}$ most similar to q_i computed by a brute-force method exhaustively searching $\mathcal{D}$. If $d_i = d_i{}^*$, the A-kNN algorithm can identify the document of $\mathcal{D}$ with the highest similarity to q_i.

Given the retrieval task addressed with A-kNN, we can measure the *recall* of a result set A-$k\text{NN}(q_i, \mathcal{D})$ of k denoted by $R^*@k$ or $R@k$, respectively. The former is computed against the *exact result set* $k\text{NN}(q_i, \mathcal{D})$, whereas the latter by considering the set of *relevant documents* for a query as identified by human assessors. Specifically, we denote by $R^*@1(q_i)$ the recall at cutoff 1 using the exact kNN algorithm, i.e., $R^*@1(q_i) = 1.0$ if $d_i = d_i{}^*$, and 0 otherwise. Also, we simply denote by $R^*@1$ the average recall, i.e., $\sum_i R^*@1(q_i)/|\mathcal{Q}|$.

A good practice in A-kNN algorithms is to set N as the minimum number of probes so that $R^*@1 \geq \rho$, $\rho \in [0, 1]$ [19]. For example, $\rho = 0.95$ means that at least 95% of the queries of $\mathcal{Q}$ returns $d_i{}^*$ as the most similar document to q_i. We can surely fix a very large value of N such that $R^*@1 = 1.0$, but this would severely hinder query processing efficiency. To make the A-kNN algorithm efficient without hampering too much effectiveness, we now discuss the baseline by Li *et al.* [19] and our proposed techniques to make A-kNN *adaptive* for the different queries $q_i \in \mathcal{Q}$. We call this family of techniques Adaptive Approximate kNN methods, as they visit the clusters *ordered* by similarity to a query q_i, but adaptively choose, for each query, the best number of clusters to probe. The hyper-parameter N of A-kNN is still used in the adaptive techniques as an upper bound of the number of clusters to visit by any given query.

Regression Approach. The regression-based approach proposed by Li *et al.* [19], which we call Reg, adaptively estimates the number $r(q_i)$ of probes to process a query q_i, where $r(q_i) \leq N$. To train and test the regression model, the query set $\mathcal{Q}$ is subdivided into train/validation/test partitions, and a golden standard $C(q_i)$ is associated with each q_i, where $C(q_i) \leq N$ is the minimum number of clusters to probe to find its closest vector $d_i{}^* \in \mathcal{D}$. If $d_i{}^*$ cannot be found in the N closest clusters, $C(q_i) = N$. It is worth noting that some features needed by the regression model depend on the k neighbors obtained after a partial search; these features are reported in groups (1), (2), and (3) of Table 1. To this end, besides N, we introduce the positive parameter τ, with $\tau \ll N$, that controls the number of clusters to visit by all queries to extract the features. Generally, small values of τ would improve the overall efficiency at the expense of the effectiveness of the predictions. Due to the imbalance of the dataset, characterized by a large fraction of queries that need to probe very few clusters, the learned regression model struggles to accurately predict the value of $C(q_i)$.

Patience-based Approach. The notion of *patience* has been long and widely explored for many tasks, primarily to prevent overfitting during model training [6, 23, 26, 38] and for early termination of inference [9, 10, 31, 32, 40–42]. We now discuss how to make use of a *patience*-based method to early-terminate A-kNN. During the iterative visit to the N clusters sorted by similarity to query q_i, the $k\text{NN}(q_i, \mathcal{D})$ result set is progressively updated. Let $RS_h(q_i)$ be the result set of the k documents obtained after visiting the first h clusters. Let $\phi_h(q_i)$ be the size, expressed in *percentage*, of the intersection between $RS_h(q_i)$ and the previous results set at iteration $h-1$, i.e., $\phi_h(q_i) = 100 \cdot |RS_{h-1}(q_i) \cap RS_h(q_i)|/k$.

Figure 1 (left) plots ϕ_h, which is the mean $\phi_h(q_i)$ for all the query $q_i \in \mathcal{Q}$. After around 30 clusters, the average ϕ_h saturates and quickly approaches 100%. This saturation phenomenon suggests we can stop the visit of further clusters when the result set does not change by much for a given number Δ of iterations, e.g., when $\phi_h(q_i) \geq \Phi \in [90, 100]$ for Δ consecutive iterations. In other words, we stop the evaluation of q_i if, for Δ consecutive iterations,

visiting the next cluster keeps at least Φ% of the k closest documents unchanged. The effectiveness of this heuristic still depends on the cluster h on which we stop the iterative search, which hopefully should be equal to or slightly greater than $C(q_i)$.

Classification Approach. To introduce this further technique, we briefly discuss the *frequency distribution* of the labels $C(q_i)$. Independently of the dense encoders adopted, we can observe that $C(q)$ follows a *power-law* distribution, where approximately 50% of all the queries in Q need to explore just the closest cluster to find and return their nearest neighbors, i.e., for about 50% of all the queries we have that $C(q_i) = 1$. Moreover, about 80% of the queries $q_i \in Q$ only need to probe at most 10 clusters to find ${d_i}^*$.

From this analysis, since most queries need only a handful of clusters to retrieve their nearest neighbor document, we could take advantage of a *classifier* aimed to early identify those queries. To this end, we reuse τ, to stop at the τ^{th} cluster the processing of all queries q_i for which $C(q_i) \leq \tau$, and proceed until N for the others. To prepare the training/validation/test dataset for the classifier, we thus need to relabel the queries: a query q_i is thus labeled as *Exit* if $C(q_i) \leq \tau$, and as *Continue* otherwise. Since the queries labeled as *Exit* form the majority class, we rebalance the two classes using SMOTE [7].

Figure 1 (right) plots the average value ϕ_h as a function of the cluster visited. Indeed, we plot two curves, each relative to the queries labeled either *Exit* or *Continue* when $\tau = 10$. We observe how ϕ_h soon becomes saturated, approaching 100% in both cases. Still, the curves are well separated. Also, on average, the *Exit* curve shows an earlier saturation than the other and is more stable as it has less variability. This also suggests that *patience*-based features can be effective for our classification task. As such, we added these features to the ones described by Li *et al.* [19] to train the classification and the regression models. The full set of features is detailed in Table 1.

Cascade Approach. If a pure classifier can successfully detect which queries have to early exit at the τ^{th} cluster, the natural follow-up question would be about what to do with the remaining *Continue* queries. A possible answer is a *cascade* technique, aimed to early stop the processing of these queries possibly before the N^{th} cluster. Specifically, we employ either a regression-based or a patience-based method for this second cascade stage, where the first stage is the classifier. In this framework, note that between *False Exit* and *False Continue*, only the former can affect effectiveness since the classifier stops processing q_i even though ${d_i}^*$ has not yet met. The latter only reduces efficiency, as the processing of q_i is not stopped and only ends when all the N clusters have been explored. When we train the classifier, we are more interested in reducing the *False Exits* as not to penalize effectiveness. So, in addition to using SMOTE, we also increase the training penalty of a *False Exit* by weighting by a factor $w \geq 1$ the instances of the *Exit* class. This weighting strategy is particularly advantageous within a cascade approach, where a large amount of *False Continues* can be early-stopped by the next cascade stage.

3 Experimental Evaluation

The experiments discussed in this section aim to answer the following research questions:

Table 1: Features used by the learned methods. Reg [19] uses groups (1), (2), and (3), while Reg+*int* and the Classifier employ all the features.

Type	Feature	Description
Query(1)	768 query values	the query vector
Centroid(2)	$\sigma(q, c_h)$ $h \in \{1..\tau\} \cup \{10, 20, ..., 100\}$	similarities of query to h-th closest centroid
Result after τ clusters(3)	$\sigma_\tau(q, d_1)$	max doc. similarity
	$\sigma_\tau(q, d_k)$	k-th doc. similarity
	$\sigma_\tau(q, d_1)$ / $\sigma_\tau(q, d_k)$	ratio of max and k-th doc. similarities
	$\sigma_\tau(q, d_1)$ / $\sigma(q, c_1)$	ratio of similarities of closest doc. and centroid
Stability(4)	$\lvert RS_{h-1}(q_i) \cap RS_h(q_i)\rvert/k$ $h \in \{2, ..., \tau\}$	intersections between consecutive results
	$\lvert RS_1(q_i) \cap RS_h(q_i)\rvert/k$ with $h \in \{2, ..., \tau\}$	intersections with first result

RQ1: Does a heuristic *patience*-based method differ significantly from a learned regression-based one?
RQ2: Can a *cascade* method improve single-stage approaches?

Experimental Settings. We experiment on the public MS-MARCO (MAchine Reading COmprehension) Passage (ver. 1) dataset [24]. We encode documents and queries in one 768-dimensional dense vector using STAR [39], CONTRIEVER [14], and TAS-B [12], generating a single embedding for each document and query in the collection. In this way, we build three collections of ~8.8M vectors. We divide the 101,093 queries into three sets to train our models. For testing, we use the official 6,980 judged subset given by the MS-MARCO maintainers[1], whereas for training and validation we divide the remaining 94,113 queries into training (67%) and validation set (33%). We use FAISS [15] to efficiently index and retrieve passages encoded as dense vectors. Specifically, we build three IVF two-level indexes to partition the collections in 65,536 clusters[2] each, based on the inner product between vectors.
Similar to Li *et al.* [19], we build our regression and classification models using small additive forests of 100 trees using LightGBM [17] and use HyperOPT [2] for hyperparameter tuning with an early stopping window set to 10.
In particular, we modified FAISS 1.7.4 and used as-is openBLAS 0.3.26 and LightGBM 4.3.0. All our experiments were performed on a machine with 504 GB of memory and two Intel Xeon Platinum 8276L CPU @ 2.20GHz with 56 physical cores. To ensure accurate measurement of the execution time, we conduct each experiment 6 times in a row, discard the initial run, and then calculate the average execution time as the average of the remaining 5 runs.

Parameter Selection. Due to space considerations, we forego a comprehensive discussion of all the parameter selections and present a brief summary of the process followed to tune them. First, for all techniques, we set the parameter k, i.e., the size of the result set, to 100. We tuned parameter τ in $\{2, 5, 8, 10, 12, 15\}$ for the classifier and the regression model. The value $\tau = 10$ consistently provides a good

[1] *dev/small*: ir-datasets.com/msmarco-passage.html#msmarco-passage/dev
[2] We use the smallest power of two greater than $16 \times \sqrt{|\mathcal{D}|} = 16 \times \sqrt{\sim 8.8\text{M}}$.

trade-off between effectiveness and efficiency. Then, to compare the methods, we align our proposals' effectiveness to that obtained by Reg [19]; we do so by choosing our parameters to minimize the scoring time and, at the same time, match the R^*@1 obtained by Reg. For example, considering STAR, Reg achieves R^*@1 = 0.93. Thus, we select our parameters to obtain the lowest execution times that show a R^*@1 $\geq$ 0.93. To penalize the prediction of *False Exits*, we increase the instance weight w of the *Exit* class during the classifier's training. We tried with 1, the default option, 3, 5, and 7, and observed that 3 permits to match Reg with STAR and TAS-B, while for CONTRIEVER we use 7. We evaluated our patience-based strategy by trying different values of Δ and Φ, i.e., $\Delta \in \{5, 7, 10, 12, 14\}$, and $\Phi \in \{90\%, 95\%, 100\%\}$. Ultimately, we chose $\Delta = 7$ for STAR, $\Delta = 12$ for CONTRIEVER, and $\Delta = 14$ for TAS-B. Finally, we set the tolerance $\Phi = 95\%$ for all of them.

Discussion. The experimental results are reported in Table 2, subdivided into three blocks, each associated with a different encoder (STAR, CONTRIEVER, and TAS-B). The first two rows of each block report the results obtained by the two baselines, namely A-kNN and the Adaptive A-kNN using Reg [19]. Following the methodology of Li *et al.* [19], we choose the minimum value of N that allows A-kNN to achieve a given R^*@1: we chose R^*@1 = 95% and thus denote the strategy by A-kNN$_{95}$. For example, considering STAR, A-kNN$_{95}$ reaches R^*@1 = 95% when all queries in Q are processed by always exploring the $N = 80$ closest clusters. The metrics R@100 and mRR@10 refer instead to the MSMARCO relevant passages, thus estimating the "real" effectiveness of the ranked result set. Finally, $\widehat{C}$ and T are the per-query average number of clusters probed and measured execution times (in *ms*), whereas Sp is the speedup obtained over the baseline A-kNN$_{95}$. In the third row of each table block, we report the results of an additional baseline denoted as Reg$_{+int}$. Reg$_{+int}$ is obtained by adding to the feature set of Reg the ones related to the stability of the result set, i.e., the features based on the intersection of the result sets inspired by our *patience*-based heuristic technique. As regards **RQ1**, the *patience*-based heuristic technique, denoted by Patience$_{\Delta=x}$, does not significantly differ from the Reg-based competitors in terms of effectiveness. However, our technique is much more efficient, with speedups ranging from 2.95× to 5.13×. The shorter processing times compared to both the Reg-based approaches are primarily due to the fewer clusters probed, as Patience$_{\Delta=x}$ visits, on average, between 35 and 84 fewer clusters per query, depending on the Reg version and encoder used. As expected, the classifier-based approach, where the training instances are weighted, can reduce the *False Exits* and is non-significantly worse than A-kNN$_{95}$ for all three encoders.

Finally we answer **RQ2**, by observing that a cascade method can increase, sometimes substantially, the efficiency of a non-cascade one. In particular, we compare a pure Reg$_{+int}$ and a pure *patience*-based early exit method with the corresponding cascade ones. For example, if we consider STAR, the relative speedups of the two cascade methods over the non-cascade ones are 1.89× and 1.39×. Analogously, with CONTRIEVER, we observe that the relative speedups of the cascade method over a pure Reg$_{+int}$ and our *patience*-based strategy are 2.19× and 1.52×. Conversely, with TAS-B, we observe speedups of only 1.26× and 1.10×. However, we observe that the effectiveness of cascade methods generally degrades, thus becoming significantly worse than A-kNN$_{95}$, with the only exception being the cascade using Reg$_{+int}$ on TAS-B. In conclusion, the more effective encoders CONTRIEVER and TAS-B need a large value of N to allow A-kNN to reach R^*@1 = 0.95. Thus, both can benefit greatly from adaptive methods that permit to inspect, on average, much less than N clusters. For these two encoders, a pure *patience*-based heuristic can achieve speedups of 4.04× and 5.13×, getting a value of mRR@10 comparable to the one obtained by A-kNN$_{95}$, while the corresponding cascade method further improves the speedup at the cost of lower effectiveness.

Table 2: Effectiveness/efficiency results. Statistical significance on mRR@10 for the EE strategy against the corresponding A-kNN$_{95}$ is tested with a pairwise t-test with Bonferroni correction (* and ** indicate p-values of < 0.05 and < 0.01).

		R^*@1	R@100	mRR@10	T (*ms*)	$\widehat{C}$	Sp
STAR ($N = 80$)	A-kNN$_{95}$	0.951	0.791	0.296	0.835	80.0	1.00
	Reg [19]	0.932	0.769	0.291	0.737	69.6	1.13
	Reg$_{+int}$	0.934	0.769	0.291	0.616	53.9	1.36
	Patience$_{\Delta=7}$	0.933	0.772	0.291	0.283	18.7	2.95
	Classifier	0.916	0.751	0.284**	0.338	25.1	2.47
	Classifier$_{w=3}$	0.930	0.763	0.289	0.305	24.2	2.74
	+ Reg$_{+int}$	0.922	0.756	0.287*	0.325	23.5	2.57
	+ Patience$_{\Delta=7}$	0.918	0.753	0.286*	0.203	12.2	4.11
CONTRIEVER ($N = 140$)	A-kNN$_{95}$	0.950	0.834	0.316	1.392	140.0	1.00
	Reg [19]	0.933	0.811	0.310	1.178	118.6	1.18
	Reg$_{+int}$	0.939	0.817	0.313	0.969	97.9	1.44
	Patience$_{\Delta=12}$	0.933	0.812	0.310	0.345	23.1	4.04
	Classifier	0.911	0.787	0.302**	0.354	25.9	3.93
	Classifier$_{w=7}$	0.939	0.819	0.311	0.522	52.8	2.67
	+ Reg$_{+int}$	0.930	0.807	0.308*	0.442	36.9	3.15
	+ Patience$_{\Delta=12}$	0.925	0.801	0.306*	0.227	13.6	6.13
TAS-B ($N = 190$)	A-kNN$_{95}$	0.951	0.826	0.323	2.027	190.0	1.00
	Reg [19]	0.920	0.796	0.315	1.655	123.1	1.22
	Reg$_{+int}$	0.928	0.799	0.316	1.265	111.9	1.60
	Patience$_{\Delta=14}$	0.921	0.798	0.314	0.395	28.3	5.13
	Classifier	0.905	0.773	0.306**	0.635	49.2	3.19
	Classifier$_{w=3}$	0.926	0.792	0.315	1.207	118.5	1.68
	+ Reg$_{+int}$	0.915	0.780	0.312	1.007	81.9	2.01
	+ Patience$_{\Delta=14}$	0.903	0.772	0.307*	0.356	22.1	5.69

4 Conclusions

This work addresses adaptive A-kNN in the context of dense, single-representation retrieval. We presented a simple, fast, and adaptable heuristic method based on the concept of patience that can achieve the same effectiveness as learned regression-based approaches with lower evaluation times and higher speedups. In future work, we will explore the behavior of clustered indexes when reducing the approximation tolerance with respect to an exact, exhaustive search. We observe that as we increase the number of clusters, the margin between our *patience* approach and the Reg-based approaches increases, and it is not clear if that is due entirely to the different encoders we considered or if it is a more general pattern.

Acknowledgements. This work was partially supported by the EU RIA project EFRA (Grant 101093026), and by the following Next Generation EU (EU-NGEU) projects: SERICS (Grant NRRP M4C2 Inv.1.3 PE00000014), iNEST (Grant NRRP M4C2 Inv.1.5 ECS00000043), and FAIR (Grant NRRP M4C2 Inv.1.3 PE00000013).

References

[1] Andoni, A., Indyk, P.: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. Communications of the ACM **51**(1), 117–122 (2008)

[2] Bergstra, J., Komer, B., Eliasmith, C., Yamins, D., Cox, D.D.: Hyperopt: a python library for model selection and hyperparameter optimization. Computational Science & Discovery **8**(1), 014008 (2015)

[3] Busolin, F., Lucchese, C., Nardini, F.M., Orlando, S., Perego, R., Trani, S.: Learning early exit strategies for additive ranking ensembles. In: Diaz, F., Shah, C., Suel, T., Castells, P., Jones, R., Sakai, T. (eds.) SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021. pp. 2217–2221. ACM (2021). https://doi.org/10.1145/3404835.3463088, https://doi.org/10.1145/3404835.3463088

[4] Busolin, F., Lucchese, C., Nardini, F.M., Orlando, S., Perego, R., Trani, S.: Early exit strategies for learning-to-rank cascades. IEEE Access **11**, 126691–126704 (2023). https://doi.org/10.1109/ACCESS.2023.3331088

[5] Cambazoglu, B.B., Zaragoza, H., Chapelle, O., Chen, J., Liao, C., Zheng, Z., Degenhardt, J.: Early exit optimizations for additive machine learned ranking systems. In: Proc. WSDM. pp. 411–420. ACM (2010)

[6] Caruana, R., Lawrence, S., Giles, C.: Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In: Leen, T., Dietterich, T., Tresp, V. (eds.) Advances in Neural Information Processing Systems. vol. 13. MIT Press (2000), https://proceedings.neurips.cc/paper_files/paper/2000/file/059fdcd96baeb75112f09fa1dcc740cc-Paper.pdf

[7] Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: Smote: synthetic minority over-sampling technique. Journal of artificial intelligence research **16**, 321–357 (2002)

[8] Chierichetti, F., Panconesi, A., Raghavan, P., Sozio, M., Tiberi, A., Upfal, E.: Finding near neighbors through cluster pruning. In: Proceedings of the Twenty-Sixth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems. p. 103–112. PODS '07, Association for Computing Machinery, New York, NY, USA (2007). https://doi.org/10.1145/1265530.1265545, https://doi.org/10.1145/1265530.1265545

[9] Gao, X., Liu, Y., Huang, T., Hou, Z.: Pf-berxit: Early exiting for bert with parameter-efficient fine-tuning and flexible early exiting strategy. Neurocomputing **558**, 126690 (2023). https://doi.org/https://doi.org/10.1016/j.neucom.2023.126690, https://www.sciencedirect.com/science/article/pii/S0925231223008135

[10] Gao, X., Zhu, W., Gao, J., Yin, C.: F-pabee: Flexible-patience-based early exiting for single-label and multi-label text classification tasks. In: ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 1–5 (2023). https://doi.org/10.1109/ICASSP49357.2023.10095864

[11] Hofstätter, S., Khattab, O., Althammer, S., Sertkan, M., Hanbury, A.: Introducing neural bag of whole-words with colberter: Contextualized late interactions using enhanced reduction. In: Proceedings of the 31st ACM International Conference on Information & Knowledge Management. p. 737–747. CIKM '22, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3511808.3557367, https://doi.org/10.1145/3511808.3557367

[12] Hofstätter, S., Lin, S.C., Yang, J.H., Lin, J., Hanbury, A.: Efficiently Teaching an Effective Dense Retriever with Balanced Topic Aware Sampling. In: Proc. of SIGIR (2021)

[13] Indyk, P., Motwani, R.: Approximate nearest neighbors: towards removing the curse of dimensionality. In: Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing. p. 604–613. STOC '98, Association for Computing Machinery, New York, NY, USA (1998). https://doi.org/10.1145/276698.276876, https://doi.org/10.1145/276698.276876

[14] Izacard, G., Caron, M., Hosseini, L., Riedel, S., Bojanowski, P., Joulin, A., Grave, E.: Unsupervised dense information retrieval with contrastive learning (2021). https://doi.org/10.48550/ARXIV.2112.09118, https://arxiv.org/abs/2112.09118

[15] Johnson, J., Douze, M., Jégou, H.: Billion-scale similarity search with GPUs. IEEE Transactions on Big Data **7**(3), 535–547 (2019)

[16] Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., Yih, W.t.: Dense passage retrieval for open-domain question answering. In: Proc. EMNLP. pp. 6769–6781 (2020)

[17] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu, T.Y.: Lightgbm: A highly efficient gradient boosting decision tree. In: Advances in Neural Information Processing Systems. pp. 3149–3157 (2017)

[18] Khattab, O., Zaharia, M.: ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. In: Proc. SIGIR. p. 39–48 (2020)

[19] Li, C., Zhang, M., Andersen, D.G., He, Y.: Improving Approximate Nearest Neighbor Search through Learned Adaptive Early Termination. In: Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (SIGMOD) (2020)

[20] Li, W., Zhang, Y., Sun, Y., Wang, W., Li, M., Zhang, W., Lin, X.: Approximate nearest neighbor search on high dimensional data — experiments, analyses, and improvement. IEEE Transactions on Knowledge and Data Engineering **32**(8), 1475–1488 (2020). https://doi.org/10.1109/TKDE.2019.2909204

[21] Li, W., Zhang, Y., Sun, Y., Wang, W., Zhang, W., Lin, X.: Approximate nearest neighbor search on high dimensional data — experiments, analyses, and improvement. IEEE Transactions on Knowledge and Data Engineering **32**, 1475–1488 (2016), https://api.semanticscholar.org/CorpusID:1364239

[22] Mikulík, A., Perdoch, M., Chum, O., Matas, J.: Learning a fine vocabulary. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) Computer Vision – ECCV 2010. pp. 1–14. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)

[23] Morgan, N., Bourlard, H.: Generalization and parameter estimation in feedforward nets: Some experiments. Advances in neural information processing systems **2** (1989)

[24] Nguyen, T., Rosenberg, M., Song, X., Gao, J., Tiwary, S., Majumder, R., Deng, L.: Ms marco: A human generated machine reading comprehension dataset. choice **2640**, 660 (2016)

[25] Pan, Z., Wang, L., Wang, Y., Liu, Y.: Product quantization with dual codebooks for approximate nearest neighbor search. Neurocomputing **401**, 59–68 (2020)

[26] Prechelt, L.: Early stopping-but when? In: Neural Networks: Tricks of the trade, pp. 55–69. Springer (2002)

[27] Reimers, N., Gurevych, I.: Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In: Proc. EMNLP. pp. 3980–3990 (2019)

[28] 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. p. 1747–1756. CIKM '22, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3511808.3557325, https://doi.org/10.1145/3511808.3557325

[29] Santhanam, K., Khattab, O., Saad-Falcon, J., Potts, C., Zaharia, M.: ColBERTv2: Effective and efficient retrieval via lightweight late interaction. In: Carpuat, M., de Marneffe, M.C., Meza Ruiz, I.V. (eds.) Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 3715–3734. Association for Computational Linguistics, Seattle, United States (Jul 2022). https://doi.org/10.18653/v1/2022.naacl-main.272, https://aclanthology.org/2022.naacl-main.272

[30] Singitham, P.K.C., Mahabhashyam, M.S., Raghavan, P.: Efficiency-quality tradeoffs for vector score aggregation. In: Nascimento, M.A., Özsu, M.T., Kossmann, D., Miller, R.J., Blakeley, J.A., Schiefer, K.B. (eds.) (e)Proceedings of the Thirtieth International Conference on Very Large Data Bases, VLDB 2004, Toronto, Canada, August 31 - September 3 2004. pp. 624–635. Morgan Kaufmann (2004). https://doi.org/10.1016/B978-012088469-8.50056-5, http://www.vldb.org/conf/2004/RS17P1.PDF

[31] Sponner, M., Ott, J., Servadei, L., Waschneck, B., Wille, R., Kumar, A.: Temporal patience: Efficient adaptive deep learning for embedded radar data processing. ArXiv **abs/2309.05686** (2023), https://api.semanticscholar.org/CorpusID:261696875

[32] Sponner, M., Servadei, L., Waschneck, B., Wille, R., Kumar, A.: Temporal decisions: Leveraging temporal correlation for efficient decisions in early exit neural networks. arXiv preprint arXiv:2403.07958 (2024)

[33] Weber, R., Schek, H.J., Blott, S.: A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In: Proceedings of the 24rd International Conference on Very Large Data Bases. p. 194–205. VLDB '98, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1998)

[34] Wu, X., Ma, G., Lin, M., Lin, Z., Wang, Z., Hu, S.: Contextual masked auto-encoder for dense passage retrieval. In: AAAI Conference on Artificial Intelligence (2022), https://api.semanticscholar.org/CorpusID:251594591

[35] Xin, J., Nogueira, R., Yu, Y., Lin, J.: Early exiting BERT for efficient document ranking. In: Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing. pp. 83–88. Association for Computational Linguistics, Online (Nov 2020). https://doi.org/10.18653/v1/2020.sustainlp-1.11, https://aclanthology.org/2020.sustainlp-1.11

[36] Xin, J., Tang, R., Yu, Y., Lin, J.: BERxiT: Early exiting for BERT with better fine-tuning and extension to regression. In: Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume. pp. 91–104. Association for Computational Linguistics, Online (Apr 2021). https://doi.org/10.18653/v1/2021.eacl-main.8, https://aclanthology.org/2021.eacl-main.8

[37] Xiong, L., Xiong, C., Li, Y., Tang, K.F., Liu, J., Bennett, P., Ahmed, J., Overwijk, A.: Approximate nearest neighbor negative contrastive learning for dense text retrieval. In: Proc. ICLR (2021)

[38] Yao, Y., Rosasco, L., Caponnetto, A.: On early stopping in gradient descent learning. Constructive Approximation **26**(2), 289–315 (Aug 2007). https://doi.org/10.1007/s00365-006-0663-2, https://doi.org/10.1007/s00365-006-0663-2

[39] Zhan, J., Mao, J., Liu, Y., Guo, J., Zhang, M., Ma, S.: Optimizing dense retrieval model training with hard negatives. In: Proc. SIGIR. p. 1503–1512 (2021)

[40] Zhang, Z., Zhu, W., Zhang, J., Wang, P., Jin, R., Chung, T.S.: PCEE-BERT: Accelerating BERT inference via patient and confident early exiting. In: Carpuat, M., de Marneffe, M.C., Meza Ruiz, I.V. (eds.) Findings of the Association for Computational Linguistics: NAACL 2022. pp. 327–338. Association for Computational Linguistics, Seattle, United States (Jul 2022). https://doi.org/10.18653/v1/2022.findings-naacl.25, https://aclanthology.org/2022.findings-naacl.25

[41] Zhou, W., Xu, C., Ge, T., McAuley, J., Xu, K., Wei, F.: Bert loses patience: Fast and robust inference with early exit. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems. vol. 33, pp. 18330–18341. Curran Associates, Inc. (2020), https://proceedings.neurips.cc/paper_files/paper/2020/file/d4dd111a4fd973394238aca5c05bebe3-Paper.pdf

[42] Zhu, W.: LeeBERT: Learned early exit for BERT with cross-level optimization. In: Zong, C., Xia, F., Li, W., Navigli, R. (eds.) Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). pp. 2968–2980. Association for Computational Linguistics, Online (Aug 2021). https://doi.org/10.18653/v1/2021.acl-long.231, https://aclanthology.org/2021.acl-long.231