A APPENDIX

A.1 Index Building and Search Configurations

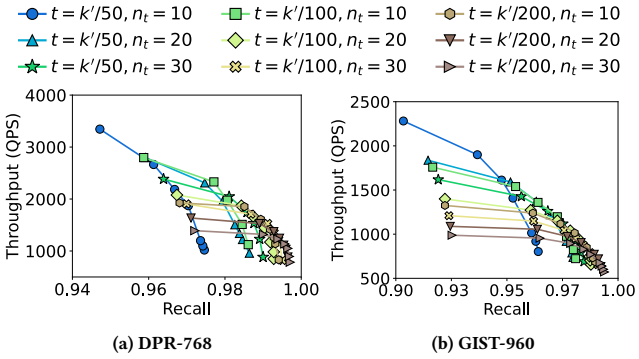
We build each index with different configurations. Then we select the best configuration for throughput-recall trade-off for high recall ≥ 0.9 to report in Figure 8. The explored configurations follow the suggestions in the respective original publications, documentation in their code repositories, or established index benchmarks [3]. Specifically, IVF-based indexes use $nlist \in \{1024, 2048, 4096\}$ and $nprobe \in \{1, 5, 10, 50, 100, 200, 300, 400, 500\}$. For PQ, $nbits = 4$ to leverage the fast scan implementation, and $m \in \{d/2, d/4, d/8\}$. For OPQ, the output dimension $d_r \in \{d, d/2, d/4, d/8\}$. ScaNN and SOAR are built with $dimensions\_per\_block = 2, T = 0.2, \#leaves \in \{1024, 2048, 4096\}$ and SOAR additionally set $overretrieve\_factor = 2$ and $\lambda = 1.5$. RabitQ uses recommended settings, $C = 4096, B_q = 4$, and searches with the same *nporbe* value range above. HNSW is built with $M \in \{16, 32, 64\}$ and $ef\_construction \in 200$, ELPIS with $max\_leaf\_size = N/5, M = 32, ef\_construction = 200$, and LSH-APG with $L = 2, K = 18, M = 24, ef\_construction = 200$. The three graph indexes use search parameters $ef\_search \in \{128, 256, ..., 4096\}$. LVQ is built with the three modes $Vamana, LVQ8, LVQ4 \times 8$, window search size 200 and $\alpha = 0.95, R = \{32, 64, 128\}$. Falconn++ use build configurations $k = 2, L \in \{100, 200, 400, 800\}, iprobe \in \{2, 4, 6, 8\}$ and search configurations $qprobe \in \{2000, 4000, 38000\}$. For LCCS, $L \in \{64, 128, ..., 2048\}$ and the search setting values are in $\{1, 2, 4, ..., 512\}$. When applicable, the refine stage with exact distance is done with $k'/k = \{1, 10, 20, 50, 100, 200, 300, 400, 500\}$. For *HAKES-Index* search optimizations, we always enable IVF centroid quantization and use $t = k'/200, n_t = 10$, even though we note tunning the early termination parameters for each dataset can yield even better throughput-recall trade-off.

The selected build configuration variable values are presented in Table 3. Indexes using one recommended build configuration and other configuration parameters using one recommended value are not included in the table.

A.2 Early Termination Parameters

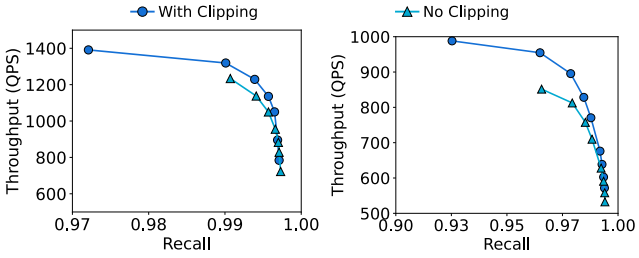
Setting the same *nprobe* to achieve high recall causes all queries to scan a large number of partitions according to the harder ones, even though some of the queries can identify all nearest neighbors in the first few partitions. As discussed in Section 3.4, *HAKES-Index* introduces an early termination checking to adapt the search process of a query based on its intermediate results, with two parameters, n_t and t. In our implementation, t is set to a value relative to k', because for a larger k' the intermediate result set keeps further away vectors and a partition adds more candidates. Figure 15 shows the effect of these two parameters on DPR-768 and GIST-960. We fix $nprobe = 200$ so that $recall > 99.5\%$ on the two datasets if query vectors are evaluated against all vectors in their closest 200 partitions with exact similarity calculation. Each curve is plotted by varying k'/k. The smaller t and n_t, the earlier the criterion is satisfied, resulting in higher throughput. However, low t and n_t can lead to queries terminating prematurely and low recall.

We then study the effect of leaving out *nprobe* and using only the termination criteria defined by t and n_t. We set $t = k'/200$ and $n_t = 30$. Figure 16, shows that if we do not clip the search



(a) DPR-768 (b) GIST-960

Figure 15: Effect of early termination parameters.



(a) DPR-768 (b) GIST-960

Figure 16: Effect of stopping at *nprobe* with early termination.

with *nprobe* the throughput-recall trade-off is worse. In *HAKES-Index* filter stage, the similarity approximation is lossy and so that partitions further away can still add candidates to the k' vector intermediate results. For some queries, n_t can only be satisfied after many more partitions than needed are scanned, leading to a drop in throughput. Therefore, *HAKES-Index* uses both criteria and terminates the query when it satisfies one.

A.3 Ablation Study on Search Optimization

Table 4 shows the impact of each search optimization at the selected configuration of each dataset that achieves $recall \approx 0.99$ with the learned parameters. The most important factor to *HAKES-Index* improved throughput-recall trade-off is the learned compression parameters. Both INT8 scalar quantization of IVF centroids and adaptive early termination provide further improvement of throughput without much recall degradation, but the benefits vary across the datasets and configurations. We used $t = k'/200$ and $n_t = 30$ for all, and the setting considerably increases the throughput on MBNET-1024, RSNET-2048, and GIST-960. As discussed in the last subsection, t and n_t can shift the throughput-recall trade-off significantly. We note here that careful tuning of them can achieve better results.

A.4 Recall Improvement on Other 1M Datasets

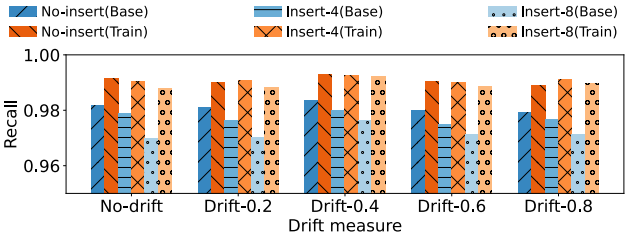
Table 5 presents the result on the remaining 1 million datasets not in Table 2. The self-supervised learning process of *HAKES-Index* still enhances the recall in the majority of the configurations in the high recall region. When *nprobe* is low, the IVF partition ranking due to

Table 3: Selected index building configuration values for index benchmarking.

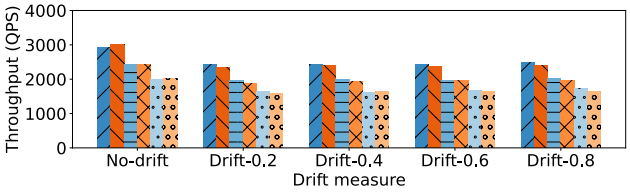
	Variables	DPR-768	OPENAI-1536	MBNET-1024	RSNET-2048	GIST-960
IVF	*nlist*	1024	1024	2048	2048	1024
IVFPQ_RF	*nlist*, *m*	1024, $d/2$	1024, $m = d/8$	1024, $d/4$	1024, $d/8$	1024, $d/2$
OPQIVFPQ_RF	*dr*, *m*, *nlist*	$d/4$, $d/2$, 1024	$d/8$, $d/2$, 1024	$d/4$, $d/2$, 2048	$d/8$, $d/2$, 1024	$d/4$, $d/2$, 1024
HNSW	M	32	32	32	16	32
ScaNN	*#leaves*	4096	4096	2048	4096	4096
SOAR	*#leaves*	4096	2048	4096	4096	4096
Falconn++	*iprobe*, *L*	16, 400	16, 100	16, 200	16, 100	8, 400
LCCS	*L*	2048	2048	2048	2048	2048
LVQ	*mode*, *R*	*LVQ*4, 64	*LVQ*4, 64	*LVQ*4, 64	*LVQ*4, 64	*LVQ*4, 64
HAKES-Index	*dr*, *m*, *nlist*	$d/4$, $d/2$, 1024	$d/8$, $d/2$, 1024	$d/4$, $d/2$, 2048	$d/8$, $d/2$, 1024	$d/4$, $d/2$, 1024

Table 4: Performance with search optimization at selected recall= 0.99 configuration.

Recall(QPS)	DPR-768	OPENAI-1536	MBNET-1024	RSNET-2048	GIST-960
All optimizations	0.990 (1276)	0.991 (1389)	0.991 (2791)	0.992 (2615)	0.988 (747)
Learned + IVFSQ	0.990 (1280)	0.994 (976)	0.991 (2579)	0.992 (2444)	0.989 (631)
Learned parameters	0.991 (1238)	0.994 (974)	0.991 (2413)	0.992 (2350)	0.989 (625)
Base parameters	0.979 (1233)	0.990 (977)	0.977 (2393)	0.982 (2330)	0.944 (610)

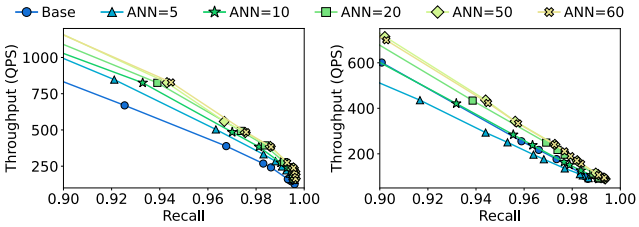


(a) Recall



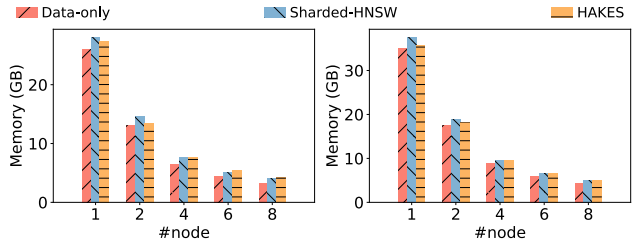
(b) Throughput

Figure 17: Tolerance against data drift (RSNET-2048).



(a) DPR-768-10m **(b) E5-1024-10m**

Figure 18: Effect of #ANN in training.



(a) DPR-768-10m **(b) E5-1024-10m**

Figure 19: Memory consumption per node.

A.5 Drift Tolerance Result on RSNET-2048

Similar to Figure 11, Figure 17 shows the tolerance of drift on the other ImageNet embeddings where we have access to the true categories. We have similar observations that the recall improvement persists across different drifts at this high recall. The addition of the new vectors with drift can impact the recall for *HAKES-Index* but not more significantly than using the base parameters for search.

A.6 Number of Near Neighbor (NN) in Learning

We plot the throughput-recall curve of Index with varying numbers of approximated nearest neighbors (NN) used in training. Figure 18 shows that increasing NN improves the performance-recall curve, until around 50. The result suggests that the training needs a sufficient number of vectors in close proximity. The intuition is that the set of candidate vectors is larger than the set of final k results. Therefore the ranking order of vectors in a larger region around the query is important to reduce the chance of missing true nearest neighbors when returning the candidate vectors.

A.7 Memory Usage in Distributed Serving

For high-dimension embedding vectors, the main storage cost comes from storing the vectors themselves. Replicating the compressed index in *HAKES* in the distributed setting has shown a

Table 5: Recall improvement at different search configurations.

IVF *nprobe* (total 1024)		10			50			100			200		
k'/k		10	50	200	10	50	200	10	50	200	10	50	200
DPR-768	Base	0.708	0.790	**0.799**	0.795	0.940	0.966	0.801	0.957	0.988	0.803	0.962	0.995
	Learned	**0.772**	**0.798**	0.798	**0.907**	**0.964**	**0.968**	**0.922**	**0.987**	**0.991**	**0.927**	**0.994**	**0.999**
OPENAI-1536	Base	0.875	0.895	**0.897**	0.940	0.969	0.972	0.952	0.983	0.987	0.957	0.990	0.994
	Learned	**0.886**	**0.897**	**0.897**	**0.956**	**0.972**	**0.974**	**0.969**	**0.986**	**0.987**	**0.975**	**0.994**	**0.995**
GIST-960	Base	0.555	0.679	**0.702**	0.645	0.867	0.936	0.651	0.884	0.967	0.652	0.888	0.975
	Learned	**0.647**	**0.689**	0.690	**0.828**	**0.932**	**0.945**	**0.847**	**0.966**	**0.984**	**0.852**	**0.975**	**0.995**

significant throughput advantage compared to the small memory overhead it causes. Figure 19 shows the average memory consumption per node with varying number of nodes. It can be seen that the memory overhead over the raw vectors is small for both *HAKES* and Sharded-HNSW across different node settings. In deployment, the number of IndexWorkers shall be approximated by dividing the overall throughput requirement by individual IndexWorker's achievable throughput and the number of RefineWorkers approximated based on the dataset size over the memory space of a single server.