QASP: Query-Adaptive Robust Vector Search Policy

Hakan Ferhatosmanoglu*
Amazon
London, UK
hakanf@amazon.com

Kushal Kumar*
Amazon
New York, USA
kushlku@amazon.com

Tal Wagner
Amazon and Tel-Aviv University
Tel Aviv, Israel
talw@amazon.com

Andy Warfield
Amazon
Vancouver, Canada
warfield@amazon.com

***Abstract*—A fundamental challenge of vector search is achieving consistently high recall while minimizing computational costs. Fixed search parameters cause significant performance variance across queries, and conventional evaluation on average recall masks these per-query disparities. We introduce QASP (Query-Adaptive robust vector Search Policy), which predicts the complete recall progression curve per query via a single upfront supervised regression, from which a search policy is derived for any recall target; this avoids iterative model invocations during search or separate predictors per target. By predicting normalized recall values with scale-invariant features and pre-search inference, QASP generalizes across recall targets, index configurations, and datasets. The fine-grained progress predictions further enable a lightweight reactive complement that adjusts search depth based on predicted-versus-observed deviations without additional inference. We prove that QASP requires a finite training sample independent of dataset size and dimensionality, that it matches or improves on the best fixed-policy loss up to a vanishing statistical error, and that its data access savings over fixed probing grow exponentially in intrinsic dimensionality. Experimentally, QASP achieves significantly lower recall variance and deviation from target, higher query satisfaction rate, and scales to large data and hierarchical indices without retraining, achieving 99% recall with 80% less data access. Our implementation can be found at https://github.com/amazon-science/QASP.**

***Index Terms*—Vector Search Policy, Approximate Nearest Neighbor Search, Proactive Policy Learning**

I. Introduction

Given a query vector $q \in \mathbb{R}^d$ and a dataset $X \subset \mathbb{R}^d$, vector (similarity) search aims to find the k vectors $\{x_1^q, \ldots, x_k^q\} \subset X$ that minimize $\text{dist}(q, x^q)$, where $\text{dist}(\cdot, \cdot)$ is a notion of distance. The fraction of k closest vectors found is the *recall* for that query. A vector search policy determines the search parameters to achieve high recall with low cost.

Practitioners typically rely on heuristic guidelines to set search parameters based on average performance, generating rules such as accessing a fixed proportion $\alpha \in [0.05, 0.1]$ of the dataset or $\lfloor\sqrt{L}\rfloor$ out of L partitions [1]–[3]. Query-agnostic settings fail to capture heterogeneous query difficulty and the resulting variation in computational needs [4]–[7]. This causes over-reading for easy queries and under-reading for difficult ones. Conventional evaluation methods compound this problem by focusing on average recall, which masks per-query disparities. Figure 1 illustrates how query-agnostic policies access significantly more data than the query-adaptive approach. Such fixed policies may require substantial over-reading to achieve the consistently high recall required by production applications, including semantic search, retrieval-augmented generation, recommendation systems, and entity resolution.

Our goal is to design a proactive search policy that consistently achieves high recall with minimal performance variance across queries regardless of difficulty. We present QASP, **Q**uery-**A**daptive robust vector **S**earch **P**olicy, which addresses this through supervised regression that predicts the complete recall progression curve, from which a policy is derived for any recall target. A single upfront inference produces the fine-grained predictions, departing from recent approaches that invoke models iteratively during search or design separate predictors per recall target. Our formulation predicts normalized recall values, and combined with scale-invariant features and pre-search inference, admits richer architectures and target-agnostic deployment. QASP's fine-grained progress predictions also enable a lightweight reactive complement that treats each progress estimate as a testable hypothesis and adjusts search depth based on deviations between predicted and observed discovery rates, without additional model inference.

QASP is designed for partitioning-based indices, e.g., Inverted File (IVF), clustering, quantization, multi-dimensional trees, due to their discrete, independent units that enable highly efficient upfront recall estimation with minimal training data, and enable proactive parameter selection and resource allocation. Partitioning-based methods are widely preferred in production systems due to their memory efficiency, parallelization across computational and storage units, compatibility with distributed architectures, and ability to achieve sublinear scaling [8].

We establish theoretical guarantees for QASP. We prove that a finite sample of queries suffices for training, independent of dataset size and dimensionality. We show a fundamental lower bound on the population loss of any fixed probe policy, determined by the variance in query difficulty, and prove that QASP matches or improves on the best fixed-policy loss up to a statistical error that vanishes as the training sample grows. This establishes generalization and competitive performance guarantees. Beyond loss, we show that QASP achieves strictly lower expected data access than fixed probing, with the savings

*Equal contribution.

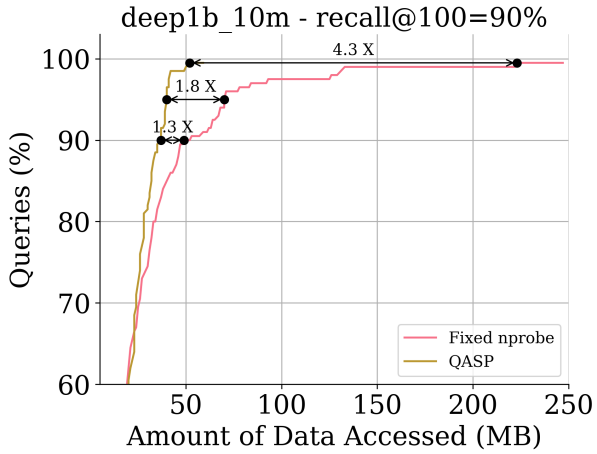


Fig. 1: Comparing fixed and adaptive policies. For fixed policy, we gradually increase the number of probes (`nprobe`) and plot query satisfaction rate (Section VI-B) versus data accessed on Deep1B 10M dataset. QASP adaptively sets per-query nprobe for the same satisfaction rate. Fixed policy consistently over-reads, with the gap widening as more queries reach recall target.

growing exponentially in the data's intrinsic dimensionality as query difficulty heterogeneity increases.

To our knowledge, this is the first work to explore transfer learning and domain adaptation in vector search. Prior solutions are mostly coupled to index configurations and recall targets. We capture scale-invariant query features and predict normalized recall values to support generalization across index configurations, recall targets, and datasets. Through ablation studies and feature importance analysis, we identify the most influential features for recall prediction.

Experimental evaluation confirms that QASP consistently outperforms baselines and achieves around 56.2% lower recall variance, 33.3% lower deviation from recall target, and 10.9% higher query satisfaction rate with similar or lower data access. QASP transfers effectively across datasets and index configurations with zero-shot or minimal fine-tuning, and scales to larger data and to hierarchical indices achieving 99% recall with 80% less data access.

To summarize, this paper makes the following contributions:

- We introduce QASP, a learning-based query-adaptive search method that predicts fine-grained recall progress, from which search policies are derived for any recall target. It enables proactive optimization beyond early termination and supports cross-domain transfer. The progress predictions also enable a lightweight monitoring mechanism that compares predicted and observed discovery rates during search and adjusts search depth accordingly.
- We establish theoretical bounds on sample size to train QASP to near optimality. We show that fixed policies leave an inherent sub-optimality gap, and prove that QASP is guaranteed to perform within this gap up to a negligible margin. We further prove that QASP achieves strictly lower expected data access, with savings growing exponentially in intrinsic dimensionality.
- We conduct experiments using both standard and query variability-aware evaluation metrics. QASP achieves significant performance improvements over baselines, with further gains on large datasets, at high recall targets, and for hard queries, where traditional approaches struggle with both consistency and cost.

II. Query-Adaptive Robust Vector Search Policy

We formalize a learning framework to design a robust search policy based on predicting fine-grained recall contributions of each index partition to optimize per-query performance. Ideal ML-enhanced search should support low inference overhead. Partitioning-based data organization offers a well-suited setting for this by decomposing search into units whose recall contributions can be predicted separately and upfront. We present our approach concretely for clustering/IVF (Inverted File) indices [9]–[13], though it extends to any partitioning-based organization, including hierarchical clustering as we study in Section II-D3. Partitioning-based indices are a core component of large-scale vector databases [12]–[16].

An IVF-type index $I(D)$ built over a dataset $D \subset \mathbb{R}^d$ is designed to reduce the computational cost of vector similarity search. To this end, it partitions D into L non-overlapping partitions (clusters) $C_1, \ldots, C_L$ with corresponding centroids $\mu_1, \ldots, \mu_L$. Given a query $q \in \mathbb{R}^d$, exhaustive search is performed only on the l top-ranked clusters, where $l \ll L$, and the clusters are ranked by their centroid distance from the query. This raises the question of how to set l, which is called the number of *probes*. Commonly, l is set to a fixed value for all queries (see Section V), which we refer to as *fixed probe policy*. Our goal is to leverage supervised learning in order to predict the optimal l per query.

To start, we formally define query-dependent probe policies.

Definition 1. *A* probe policy *is a map* $g : \mathbb{R}^d \times [0,1] \to \mathbb{N}$, *which maps a query* $q \in \mathbb{R}^d$ *and a recall target* $r \in [0,1]$ *to a number* $l = g(q,r)$ *of top-ranked clusters to probe for* q. *The policy is* valid *for an index* $I(D)$ *with* L *clusters if* $1 \leq g(q,r) \leq L$ *for all* $q \in \mathbb{R}^d$ *and* $r > 0$ *(thus, the policy instructs us to probe at least one cluster and no more than all clusters per query).*

Given a query set $Q \subset \mathbb{R}^d$ and recall target $r^* \in [0,1]$, we say that a policy g is *optimal* for Q and r^* if it probes exactly the minimal number of clusters necessary to achieve recall r^* on each query $q \in Q$. Formally, let $R_{I(D)}(q,l)$ denote the recall attained for a query q when probing its l top-ranked clusters in $I(D)$. Then, the optimal number of probes ℓ^* can be defined as follows:

$$\ell^*(q,r) := \min\{l : R_{I(D)}(q,l) \geq r\}. \tag{1}$$

An optimal policy g^* for Q and r^* satisfies $g^*(q,r^*) = \ell^*(q,r^*)$ for all $q \in Q$. This does not specify the behavior of g on queries $q \in \mathbb{R}^d \setminus Q$, i.e., it does not specify how to generalize beyond Q.

A. Ordinal Regression

Our goal is thus to find a policy which is optimal (or close to it) on Q, while also generalizing to unseen queries $q \in \mathbb{R}^d \setminus Q$ and to recall values beyond r^*. This can be naturally cast as a supervised learning problem, where we train a machine learning model to predict the behavior of the index $I(D)$ on the workload Q, using its observed performance on Q as well as the ground-truth nearest neighbors of Q in D as labels to train on. Supervised learning is useful here for two key properties: on the one hand, ML models can be efficiently trained to find high-quality solutions in the large function spaces, like that of probe policies; on the other hand, if trained with suitable features, ML models are known to generalize well to unseen inputs.

We now formalize an appropriate supervised learning problem. Let Θ be the parameter space for a parameterized class $\mathcal{G} = \{g_\theta : \theta \in \Theta\}$ of probe policies. To find an optimal probe policy in $\mathcal{G}$ for queries Q and recall target r^*, consider the optimization problem:

$$\theta^* = \arg\min_{\theta \in \Theta} \frac{1}{L^2|Q|} \sum_{q \in Q} \left(\ell^*(q, r^*) - g_\theta(q, r^*)\right)^2 . \quad (2)$$

This is an ordinal regression problem, since its goal is to learn a function g_{θ^*} whose output is an ordinal value $l \in \{1, \ldots, L\}$. The normalization by $L^2|Q|$ ensures that the minimized loss is in $[0, 1]$ for convenience and does not alter the minimization problem.

B. Recall-based Regression

Since ordinal regression is notoriously difficult to optimize [17], we develop an alternative recall-based approach for learning the probe-policy. To this end, we learn a *recall predictor* as an intermediate step. It maps a query q and number of probes l to the recall it expects to achieve for that query by probing the index its l top-ranked clusters. A recall predictor gives rise to a probe policy by using the minimal number of probes predicted to attain the recall target.

Definition 2. *Given an index $I(D)$ with L clusters, a* recall predictor *is a function $f : \mathbb{R}^d \times \mathbb{N} \to [0, 1]$ that maps a query and probe count to a predicted recall value.*

Definition 3. *A recall predictor f induces a probe policy $g^{[f]}$ defined as*

$$g^{[f]}(q, r) := \min\{l : f(q, l) \geq r\}. \quad (3)$$

Let $\widetilde{\Theta}$ parameterize a class of recall predictors, $\mathcal{F} = \{f_\theta : \theta \in \widetilde{\Theta}\}$. The induced class of parameterized probe policies is $\mathcal{G} = \{g_\theta = g^{[f_\theta]} : \theta \in \widetilde{\Theta}\}$. We learn a recall predictor by optimizing an L2 loss:

$$\theta^* = \arg\min_{\theta \in \Theta} \frac{1}{L|Q|} \sum_{q \in Q} \sum_{l=1}^{L} \left(f_\theta(q, l) - R_{I(D)}(q, l)\right)^2 . \quad (4)$$

Learning a recall predictor as a surrogate for learning a probe policy is justified by the following consistency property.

Proposition 4. *Suppose $\theta^* \in \widetilde{\Theta}$ is such that f_{θ^*} attains zero loss in eq. (4). Then $g_{\theta^*} = g^{[f_{\theta^*}]}$ attains zero loss in eq. (2).*

Proof. Zero loss in eq. (4) implies that $f_{\theta^*}(q, l) = R_{I(D)}(q, l)$ for all q, l. Plugging this in eqs. (1) and (3) yields $\ell^*(q, r) = g^{[f_{\theta^*}]}(q, r)$ for all q, r. This yields zero loss in eq. (2). □

Recall-based regression (eq. (4)) offers numerous advantages over direct ordinal regression for the probe policy (eq. (2)). First, it is more tractable to optimize [17]. Second, since its target predictions are values in a fixed range $[0, 1]$, independent of the number of clusters L, the same induced policy is transferrable across indices with different numbers of clusters. Third, the learning problem in eq. (4), unlike that in eq. (2), is independent of a recall target r^*, and thus generalizes better across different recall values. Fourth, the loss function in eq. (4) directly involves L numerical values obtained from the index $I(D)$ per query (the values $\{R_{I(D)}(q, l) : l = 1, \ldots, L\}$), compared to just one such value in eq. (2) (the value $\ell^*(q, r^*)$). This allows the learning model to use more supervised labels per query from the index, increasing ground-truth utilization and learning effectiveness. Directly predicting nprobe couples the model to a specific recall target and index size, and provides $L\times$ fewer supervision signals per query. We confirm empirically (in Section VI-D) that single-nprobe prediction results in significantly higher variance and more data access than QASP (Table II). Fifth, recall-based regression inherently produces a recall prediction per cluster, and these values can be leveraged at search time for further optimizations without model inference, such as reactive early/late termination if rate of newly discovered neighbors per probe subceeds/exceeds (respectively) the initial recall predictions (see Section II-E).

C. Model Training

QASP employs supervised learning to model the relationship between query characteristics and retrieval performance. Training data generation and model selection are as follows.

1) Training Data Generation: We obtain training signals by searching the index at different probe counts for queries sampled from the dataset, or query workload if available. We measure recall at every searched partition and use it as the prediction target. Exact k-NN search or a high-recall ANN configuration can be used to provide ground-truth labels. We obtain query features as a by-product of this process with no additional pass (see Section III). The resulting dataset is split into training and validation sets and used to fit a QASP model.

The signal-generation process can reuse the data pass performed during index construction. It can also be performed on a smaller index.

2) Model Selection: We build three variants of pre-trained QASP models. First, *QASP-DL* is a lightweight deep learning architecture for tabular learning [18] with batch normalization and dropout. Second, *QASP-GBDT* is a gradient boosted decision tree model which is a de-facto architecture for tabular learning tasks due to their ability to handle decision manifolds effectively [19]. Third, *QASP-LITE* is a polynomial regression

model. All variants are trained to optimize mean squared error loss and differ in expressive power, inference latency, ease of integration and adaptability to unseen domains.

D. Model Invocation

Training QASP models on the recall regression task serves as a surrogate for the search policy during model invocation Section II-B. Beyond the naive application of QASP to the same data distribution as it is trained on, QASP can be effectively employed at search-time.

1) Model Inference: QASP model inference is performed proactively on the query workload prior to vector search execution. This design allows rich architecture such as deep learning unlike prior work [20] due to workload-level optimizations. Batching for large-scale inference allows easy parallelism and GPU acceleration can yield significant scaling benefits. By processing queries in batches, the approach amortizes computational overhead and sustains high throughput, even as the index partitioning scales to orders of magnitude beyond the size observed during training.

2) Domain Adaptation: Recall prediction serves as a foundational training objective for vector search, generalizing effectively across datasets, query distributions, and index configurations. We design features that are normalized and scale-invariant (see Section III), which capture fundamental search difficulty signals that transfer across heterogeneous scales and indexing schemes. This enables QASP to adapt to new domains via both zero-shot transfer, where the pre-trained model generalizes to unseen datasets and index configurations without adjustment, and few-shot transfer, with limited training on batch normalization [21] and final projection layers using very few queries from the target domain.

3) Scaling to Hierarchical Indices: At invocation time, QASP can be scaled to large indices without additional training by extending the same feature transformations on hierarchical partitioning. QASP models can effectively be trained once on a smaller flat index and applied across all hierarchy levels. At inference, model features at each level are computed using coresets, with second-level centroids already available from index construction providing a lightweight approximation. This combination enables fast, scalable model invocation across large indices.

E. Lightweight Reactive Complement

QASP's fine-grained recall predictions $R_1, \ldots, R_L$ enable real-time validation as search progresses: Each prediction R_i represents cumulative recall after searching i partitions, making the incremental contribution $\widetilde{DR}_i = R_i - R_{i-1}$ a testable hypothesis about partition i's value. For example, when QASP predicts that partition i will increase recall from 0.89 to 0.94, it asserts that this partition contains approximately $0.05k$ true nearest neighbors, which can be immediately verified during search. The actual discovery rate DR_i is observed as partition i's contribution to the running result set.

The goal is to detect whether predicted rates $\widetilde{DR}_i$ under-predict or over-predict observed rates DR_i as search progresses. We adapt a lightweight statistical process control solution using EWMA smoothing: $SDR_i = \alpha \cdot DR_i + (1-\alpha) \cdot SDR_{i-1}$ to prioritize recent discovery rates. A Shewhart-type run rule detects consistent deviations $\Delta_i = (SDR_i - \widetilde{DR}_i)/\widetilde{DR}_i$ over consecutive steps. Consistently negative values indicate early termination opportunities, while consistently positive values suggest extending search beyond initial predictions. See Section VI-G for pseudocode.

III. Model Features

Model features are data signals that are informative for recall prediction task and accessible at inference with no additional computation. We define a set of features, with scale-invariant transformations, designed to capture diverse query and index characteristics.

A. Feature Types

Dataset Features. Data Size: $1/\sqrt[d]{n}$, where $n = |D|$, is the dataset size and d is the vector dimensionality. This quantity captures the expected separation (min distance) between n points packed in a unit sphere.

Index Features. Rank: $1 - \exp(-l/L)$, where l is centroid rank and L is total number of centroids. Normalization by L ensures rank $\in (0, 1 - 1/e]$ for all indices. Exponential decay captures diminishing returns of accessing further clusters.
Cumulative Cluster Size: $\sum_{i=1}^{l} |C_{\pi_q(i)}|/|D|$. This represents fraction of vectors read up to a given centroid rank l. This feature is especially useful for imbalanced indices.
Cluster Coefficient of Variance: $\sigma(d(v_l, \mu_l))/\mu(v_l, \mu_l)$, where $\sigma(d(v_l, \mu_l))$ is the standard deviation and $\mu(d(v_l, \mu_l))$ is the mean distance between data vectors $v_l \in C_l$ and centroid μ_l.
Maximum Cluster Distance: $\max(v_l, \mu_l)/d(q, \mu_{\pi_q(1)})$, where $\max(v_l, \mu_l)$ is the max distance between data vectors $v_l \in C_l$ and centroid μ_l and $d(q, \mu_{\pi_q(1)})$ is the distance between query and the centroid of nearest cluster. We also consider other statistical summaries (like $min, p25, p75$) over the distance distribution. Normalization with $(q, \mu_{\pi_q(1)})$ is performed at search time.

Query Features. Relative Distance: $\tanh(\delta(q, l) - 1)$, where $\delta(q, l) = d(q, \mu_l)/d(q, \mu_{\pi_q(1)})$ and $d(q, \mu_{\pi_q(1)})$ is the distance from query q to its nearest centroid. Query-to-centroid distances are normalized relative to the nearest centroid distance.

Hyperbolic tangent compresses distances into bounded ranges and aims to capture diminishing returns of accessing distant clusters.
Local Relative Contrast: $\mu(d(q, \mu_{\pi_q(\cdot)}))/d(q, \mu_{\pi_q(1)})$, where $d(q, \mu_{\pi_q(1)})$ is the distance from query q to its nearest centroid and the numerator represents the average distance to all other centroids. This feature is an adaptation of the local relative contrast defined by [22] using centroid vectors. We also adapt *Local Intrinsic Dimensionality* similarly using the formula $-L/\sum_{i=1}^{L}[\ln d(q, \mu_{\pi_q(i)}) - \ln d(q, \mu_{\pi_q(L)})]$, where $d(q, \mu_{\pi_q(i)})$ is the distance from query q to the ith centroid and $d(q, \mu_{\pi_q(L)})$ is the distance between q and the farthest centroid. However, this feature is ablated from the final feature set (see Section VII-C).

B. Feature Transformation

The foregoing feature design incorporates various transformations (e.g., normalization with respect to L or $|D|$, using exponential decay and hyperbolic tangent) to bound feature ranges within fixed intervals, thereby providing a scale-invariant feature representation across varying search configurations. By expressing features as normalized ratios, they remain meaningful across different vector space distributions, dataset scales or index configurations. Another transformation we consider is *relative difference* or "relative jumps" for any feature, call it $\varphi(q, l)$, varying with l:

$$\Delta(\varphi; q, l) = \frac{\varphi(q, l) - \varphi(q, l-1)}{\max_{1<j<l}(\varphi(q, j) - \varphi(q, j-1))}.$$

This allows QASP to rely not only on the raw feature value but also on how much the feature value changes as more clusters are probed in order to detect diminishing returns. We pad the output of this transformation with 0 wherever undefined. We conduct feature analysis and ablations to find the feature set (see Section VII-C).

IV. Theoretical Guarantees for QASP

In this section we analyze QASP's optimization and connect the structure of the vector index to the provable advantage of QASP. We first show that every fixed policy suffers irreducible loss determined by the variance of query difficulty, and that QASP can be trained nearly optimally with a finite number of samples independent of dataset size and never outperformed by any fixed policy except by a vanishing margin. We then derive a dominance condition proving that QASP's data access savings grow exponentially in intrinsic dimensionality.

Let $\mathscr{Q}$ be a query distribution over $\mathbb{R}^d$. Suppose that the given query set Q is a sample from $\mathscr{Q}$. Let $r^* \in [0, 1]$ be the recall target. We define the population loss of a probe policy g on $\mathscr{Q}$ and r^* as

$$\mathcal{L}(g) := \mathbb{E}_{q\sim\mathscr{Q}}\left[\frac{1}{L^2}(\ell^*(q, r^*) - g(q, r^*))^2\right], \tag{5}$$

where $\ell^*(q, r^*)$ is the minimum number of clusters in $I(D)$ that need to be probed to attain recall r^* for q, as per eq. (1). For a finite sample Q from $\mathscr{Q}$, we define the corresponding empirical loss $\hat{\mathcal{L}}(g)$ as follows. Observe that the probe policy learning problem in eq. (2) minimizes this empirical loss on Q.

$$\hat{\mathcal{L}}(g) := \frac{1}{L^2|Q|}\sum_{q\in Q}(\ell^*(q, r^*) - g(q, r^*))^2.$$

A. Sub-Optimality of Fixed Probe Policies

We first provide a suboptimality bound for the population loss of fixed policies. A fixed policy g maps all queries to a fixed number of probes. We denote by $g^{(l)}$ the fixed l-probe policy, defined as

$$\forall q \in \mathbb{R}^d, \quad g^{(l)}(q, r^*) = l. \tag{6}$$

We define a distribution over $\{1, \ldots, L\}$ from $\mathscr{Q}$. Let Λ be a random variable in $\{1, \ldots, L\}$ drawn by sampling $q \sim \mathscr{Q}$ and setting

$$\Lambda = \ell^*(q, r^*). \tag{7}$$

Thus, Λ is the optimal number of clusters that needs to be scanned for a query drawn at random from $\mathscr{Q}$, under the recall target r^*. We can relate the sub-optimality of fixed probe policies to the distribution of Λ as follows.

Theorem 5. *For every $l \in \{1, \ldots, L\}$, the population loss of the fixed-probe policy $g^{(l)}$ satisfies $\mathcal{L}(g^{(l)}) \geq \mathrm{Var}(\Lambda)/L^2$.*

Thus, the single quantity $\frac{\mathrm{Var}(\Lambda)}{L^2}$ is a uniform lower bound on the population losses of every fixed probe policy simultaneously. We can evaluate this quantity — either empirically from the given query workload, or analytically under a distributional query model — to quantify the sub-optimality enforced by restricting the search procedure to fixed-probe policies.

Example—Zipfian probes: The ρ-Zipfian law, $\Pr[\ell] \propto 1/\ell^\rho$, is commonly used to model access frequency patterns [23]. For simplicity, we focus on the most standard case $\rho = 1$. Thus, for a partition-based vector index with L clusters and a query drawn at random from $\mathscr{Q}$, the probability that its optimal number of probes is ℓ is proportional to $1/\ell$. Hence, Λ is distributed as $\Pr[\Lambda = \ell] = 1/(\ell \cdot H_L)$, where $H_L = \sum_{i=1}^{L}\frac{1}{i} = \Theta(\ln L)$ is the L^{th} harmonic number.

A direct calculation yields that

$$\mathrm{Var}(\Lambda) = \frac{L(L+1)}{2H_L} - \frac{L^2}{H_L^2} = \Theta\left(\frac{L^2}{H_L}\right) = \Theta\left(\frac{L^2}{\ln L}\right).$$

Hence, by Theorem 5, the population loss (5) of every fixed-probe policy is at least $\Omega(1/\ln L) > 0$.

Proof of Theorem 5. We recall the following basic fact:

Lemma 6. *For every real-valued random variable X with $\mathbb{E}[X] < \infty$, it holds that* $\arg\min_{\mu\in\mathbb{R}} \mathbb{E}\left[(\mu - X)^2\right] = \mathbb{E}[X]$.

Using this, every fixed probe policy $g^{(l)}$ satisfies,

$$\begin{aligned}
\mathcal{L}(g^{(l)}) &= \mathbb{E}_{q\sim\mathscr{Q}}\left[\frac{1}{L^2}(g^{(l)}(q, r^*) - \ell^*(q, r^*))^2\right] && \text{eq. (5)}\\
&= \frac{1}{L^2}\mathbb{E}_{q\sim\mathscr{Q}}\left[(l - \ell^*(q, r^*))^2\right] && \text{eq. (6)}\\
&\geq \frac{1}{L^2}\min_{\mu\in\mathbb{R}}\mathbb{E}_{q\sim\mathscr{Q}}\left[(\mu - \ell^*(q, r^*))^2\right]\\
&= \frac{1}{L^2}\mathbb{E}_{q\sim\mathscr{Q}}\left[\left(\mathbb{E}_{q\sim\mathscr{Q}}[\ell^*(q, r^*)] - \ell^*(q, r^*)\right)^2\right]\\
&= \frac{1}{L^2}\mathbb{E}\left[\left(\mathbb{E}\left[\Lambda\right] - \Lambda\right)^2\right] = \frac{\mathrm{Var}(\Lambda)}{L^2}. && \text{eq. (7)}
\end{aligned}$$

B. QASP versus Fixed Probe Policies

We show that a finite sample of queries Q from $\mathscr{Q}$ suffices to train QASP to better loss than any fixed probe policy with high probability, up to a small gap that vanishes as the sample size grows. Importantly, the requisite sample size is independent of the dataset size $|D|$, and depends only on the number of clusters L and on the structure of the QASP model.

We start with some learning-theoretic definitions. The pseudo-dimension characterizes the number of samples required to certify generalization [24]–[28]. We define pseudo-dimension for probe policies as follows:

Definition 7 (pseudo-dimension of probe policy class)**.** *Let $\mathcal{G}$ be a class of probe policies. Let $Q = \{q_1, \ldots, q_M\} \subset \mathbb{R}^d$. We say that Q is* pseudo-shattered *by $\mathcal{G}$ if there exist thresholds $\tau_1, \ldots, \tau_M \in \mathbb{R}$ such that for every $I \subset [M]$, there is $g \in \mathcal{G}$ such that $g(q_i, r^*) > \tau_i \Leftrightarrow i \in I$. The* pseudo-dimension *of $\mathcal{G}$ is denoted $\mathrm{pdim}(\mathcal{G})$ and defined as the largest size of a set $Q \subset \mathbb{R}^d$ pseudo-shattered by $\mathcal{G}$.*

Function classes with finite pseudo-dimension are called *learnable* because their population loss can be approximately optimized from a finite sample. Specifically, the pseudo-dimension governs the sample size required to guarantee that the empirical loss approximates the population loss for every probe policy in the class, a property known as *uniform convergence* [29].

Theorem 8 ([30], Theorem 19.2)**.** *Let $\varepsilon, \eta \in (0,1)$. Let Q be a sample from $\mathscr{Q}$ of size $O((\mathrm{pdim}(\mathcal{G}) + \log(1/\eta))/\varepsilon^2)$. Then,*

$$\Pr\left[\forall g \in \mathcal{G} : \ |\mathcal{L}(g) - \hat{\mathcal{L}}(g)| < \varepsilon\right] \geq 1 - \eta.$$

The pseudo-dimension can be bounded using the dimension of the parameter space and the complexity of the learned model, as per the following theorem, which is a specialization of Theorem 8 from [31] (based on [32], [33]) to probe policies.

Theorem 9. *Let $\mathcal{G} = \{g_\theta : \theta \in \Theta\}$ be a class of probe policies parameterized by an m-dimensional parameter space Θ. Let $F(q)$ be any set of numerical features representing a query $q \in \mathbb{R}^d$. Suppose that for every $\theta \in \Theta$ and $q \in \mathbb{R}^d$, it is possible to compute $g_\theta(q, r^*)$ from θ and $F(q)$ with an algorithm that can perform arithmetic operations $\{+, -, \times, \div\}$ and numerical comparisons $\{=, \neq, >, \geq, <, \leq\}$. Suppose the algorithm has running time t. Then, $\mathrm{pdim}(\mathcal{G}) = O(mt)$. Furthermore, if the algorithm is also allowed to perform exponentiation ($x \mapsto e^x$), then $\mathrm{pdim}(\mathcal{G}) = O((mt)^2)$.*

We show that our implicit ML-based search policy, where probe decisions are induced through recall predictions can be analyzed through bounded pseudo-dimension theory. We prove that QASP's compositionally-defined policy class has finite pseudo-dimension despite complex interactions between neural network recall predictors, query features, and the minimization operation that determines optimal probe counts.

Lemma 10. *The pseudo-dimension $\mathrm{pdim}(\mathcal{G}_{\mathrm{QASP}})$ of all three QASP variants is finite. Furthermore, it depends only on L, and is independent of the dataset size n and the dimensionality d.*

Proof. Invoke Theorem 9. By Section III, the number of features $|F(q)|$ per query q is $O(L)$, hence so is the number of trainable parameters (i.e., the dimension of the parameter space) per QASP model. By Section II-C2, all three QASP architectures can be computed from $F(q)$ using only arithmetic operations and numerical comparisons, and (in the case of QASP-DL) exponentiations (in order to compute sigmoid activations). Therefore, the proposition follows from Theorem 9. □

Theorem 11. *Let $\varepsilon, \eta \in (0,1)$. Suppose the sample Q has size $O(\mathrm{pdim}(\mathcal{G}_{\mathrm{QASP}})/\varepsilon^2 + \log(1/\eta))$. Let g_{QASP} be the policy that minimizes eq. (2). Then, with probability $1-\eta$, it holds that*

$$\forall\, l \in [L], \quad \mathcal{L}(g_{\mathrm{QASP}}) \leq \mathcal{L}(g^{(l)}) + 2\varepsilon. \tag{8}$$

Note that both the failure probability η and the performance gap 2ε vanish as we increase the sample size $|Q|$.

Proof. With Lemma 10, we can prove Theorem 11. Since g_{QASP} minimizes the empirical loss $\hat{\mathcal{L}}$, we have,

$$\forall\, l \in [L], \quad \hat{\mathcal{L}}(g_{\mathrm{QASP}}) \leq \hat{\mathcal{L}}(g^{(l)}). \tag{9}$$

By definition, the learned policy g_{QASP} is realizable in the QASP model, meaning $g_{\mathrm{QASP}} \in \mathcal{G}_{\mathrm{QASP}}$. Furthermore, all fixed-probe policies $g^{(l)}$ are realizable in the QASP model. Therefore, by Lemma 10 and Theorem 8, we have with probability $1-\eta$,

$$|\mathcal{L}(g_{\mathrm{QASP}}) - \hat{\mathcal{L}}(g_{\mathrm{QASP}})| < \varepsilon \ \text{ and } \ \forall\, l, |\mathcal{L}(g^{(l)}) - \hat{\mathcal{L}}(g^{(l)})| < \varepsilon.$$

Together with eq. (9), with probability $1-\eta$, eq. (8) follows. □

Remark. The same conclusion as Theorem 11 holds if we define the empirical loss with eq. (4) instead of eq. (2) and the appropriate analogous population loss (replacing $\Sigma_{q \in Q}$ with $\mathbb{E}_{q \sim \mathscr{Q}}$ in eq. (4)). The formal proof goes by augmenting the feature set $F(q)$ with the additional query features $\{R_{I(D)}(q, l)\}_{l=1}^{L}$, which enables computing the contribution of q to the loss from $F(q)$ while maintaining the size bound $|F(q)| = O(L)$. We omit further details.

In Section IV-A we establish that fixed policies inherently leave room for loss improvement.

Lemma 10 allows us to complement this result and prove that even in the worst case, with high probability, QASP's loss is never *worse* than any fixed policy, except by a negligible margin that tends to zero as the sample size grows.

C. Cost Dominance of Adaptive Probing

Section IV-A and Section IV-B establish that QASP generalizes from finite samples and that fixed policies are inherently sub-optimal in loss. We now ask a complementary question: *how much data access does QASP save*, and under what conditions does it dominate fixed probing despite predictor imperfection? Fixed policies must provision for the tail of the query difficulty distribution, forcing every query, including easy ones, to pay the cost of the hardest queries. QASP instead pays what each query individually requires. We formalize this intuition and derive the precise conditions under which QASP dominates, connecting the geometry of the vector index to the quality of the learned predictor, with the gap growing exponentially in the intrinsic dimensionality of the data.

We introduce three quantities characterizing the index. Let $D_I = \max_j \max_{x \in C_j} \|x - \mu_j\|$ be the maximum cluster radius, $\sigma_I = \min_{j \neq j'} \|\mu_j - \mu_{j'}\|$ the minimum inter-centroid separation, and Δ the doubling dimension [1] of the dataset union centroids. Write $r(q) = \|q - x^*(q)\|$ for the nearest-neighbor distance of query q, F_r for its CDF over $\mathcal{Q}$, and $r_\delta = F_r^{-1}(1-\delta)$ for the $(1-\delta)$-quantile.

Lemma 12 (Cell Intersection Bound)**.** $\ell^*(q,1) \leq \big(4(r(q)+D_I)/\sigma_I\big)^\Delta$.

Proof. Let j^* be the cluster containing $x^*(q)$. By the triangle inequality, $\|q-\mu_{j^*}\| \leq r(q)+D_I$, so $\mu_{j^*} \in B(q, r(q)+D_I)$. Every centroid ranked before μ_{j^*} also lies in this ball, so $\ell^*(q,1)$ is at most the number of centroids in $B(q, r(q)+D_I)$. Since centroids are pairwise separated by $\geq \sigma_I$, the balls $\{B(\mu_j, \sigma_I/2)\}$ are disjoint and contained in $B(q, r(q)+D_I+\sigma_I/2)$. By the doubling-dimension packing property [34], a ball of radius R contains at most $(2R/\rho)^\Delta$ points with pairwise distance $\geq \rho$. Setting $R = r(q)+D_I+\sigma_I/2$ and $\rho = \sigma_I$ gives $\ell^*(q,1) \leq (2(r(q)+D_I)/\sigma_I + 1)^\Delta \leq (4(r(q)+D_I)/\sigma_I)^\Delta$ when $r(q)+D_I \geq \sigma_I/2$. □

The ratio $(r(q)+D_I)/\sigma_I$ captures geometric query difficulty: queries far from their nearest neighbor or in poorly separated partitions require more probes. This bound is tight up to constants: by definition of doubling dimension, there exist point configurations where $\Theta((R/\rho)^\Delta)$ points with pairwise separation $\geq \rho$ fit in a ball of radius R [34], so the upper bound is achievable. Note that $\ell^*(q,1)$ bounds the probes needed to retrieve the single nearest neighbor; for k-NN recall at target r^*, the bound is conservative since achieving partial recall is strictly easier.

Proposition 13 (Fixed Probe Requirement)**.** *A fixed-ℓ policy achieves recall $\geq 1-\delta$ when*

$$\ell \geq \ell_{\text{fix}}(\delta) := \left(\frac{4(r_\delta + D_I)}{\sigma_I}\right)^\Delta. \tag{10}$$

Proof. By Lemma 12, $r(q) \leq \frac{\sigma_I}{4}\ell^{1/\Delta} - D_I$ implies $\ell^*(q,1) \leq \ell$. Setting this threshold $\geq r_\delta$ gives (10). □

This is the cost every query pays under fixed probing — including easy queries with $r(q) \ll r_\delta$ that would succeed with far fewer probes. We now bound what QASP pays instead. Let $\hat{\ell}(q) = g_{\text{QASP}}(q, r^*)$ be QASP's predicted probe count. Define the *failure rate* $\varepsilon = \Pr[\hat{\ell}(q) < \ell^*(q,r^*)]$ and the *overshoot* $\bar{\eta} = \mathbb{E}[\hat{\ell}(q) - \ell^*(q,r^*) \mid \hat{\ell}(q) \geq \ell^*(q,r^*)]$. The failure rate ε is the probability QASP under-probes (missing the recall target); the overshoot $\bar{\eta}$ is the average wasted probes when it succeeds.

Proposition 14 (QASP Expected Probes)**.** $\mathbb{E}[\hat{\ell}(q)] \leq \mathbb{E}[\Lambda] + \bar{\eta} + \varepsilon \cdot L$.

Proof. Condition on success ($\hat{\ell} \geq \ell^*$) and failure ($\hat{\ell} < \ell^*$): $\mathbb{E}[\hat{\ell}] = (1-\varepsilon)(\mathbb{E}[\ell^* \mid \text{success}] + \bar{\eta}) + \varepsilon\,\mathbb{E}[\hat{\ell} \mid \text{failure}]$. Since $(1-\varepsilon)\mathbb{E}[\ell^* \mid \text{success}] \leq \mathbb{E}[\Lambda]$, $(1-\varepsilon)\bar{\eta} \leq \bar{\eta}$, and $\mathbb{E}[\hat{\ell} \mid \text{failure}] \leq L$ (a worst-case bound; in practice failure queries cost far less than L probes), the bound follows. □

[1] The smallest integer such that every ball of radius r is covered by 2^Δ balls of radius $r/2$.

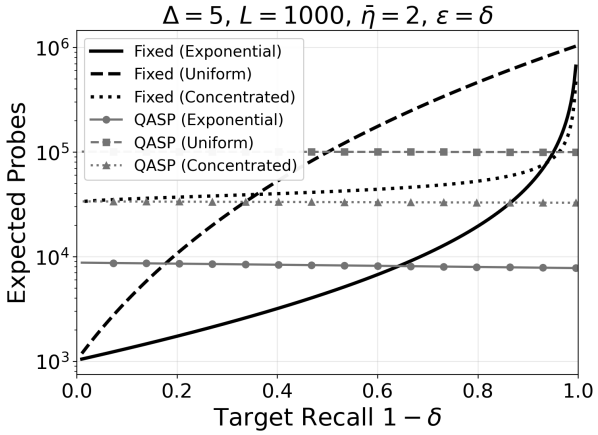


Fig. 2: Expected probes vs. recall target for fixed probing (black) and QASP (gray) under three NN distance distributions ($\Delta = 5$, $L = 1000$, $\bar{\eta} = 2$, $\varepsilon = \delta$). The vertical gap between matched curves is the data access savings from adaptive probing. QASP benefits most when F_r has a heavy tail (exponential, uniform) and at high recall targets, where fixed probing must provision for the worst-case quantile r_δ.

Corollary 15 (Dominance Condition)**.** *At matched recall ($\varepsilon = \delta$), QASP has strictly lower expected data access than fixed probing whenever*

$$\underbrace{\bar{\eta} + \varepsilon \cdot L}_{\text{predictor overhead}} < \underbrace{\left(\frac{4(r_\delta + D_I)}{\sigma_I}\right)^\Delta - \mathbb{E}[\Lambda]}_{\text{tail waste of fixed probing}}. \tag{11}$$

Combining Propositions 13 and 14 yields the main result. The left side is the total cost of predictor imperfection: wasted probes from over-prediction ($\bar{\eta}$) plus the penalty from under-prediction failures (εL). The right side is the *tail waste*: the gap between the fixed probe count (driven by the hardest queries via r_δ) and the average difficulty $\mathbb{E}[\Lambda]$. This gap grows as the Δ-th power of the ratio between worst-case and average query difficulty. QASP dominates whenever the predictor is "good enough" relative to this heterogeneity; conversely, when F_r is concentrated, all queries are equally hard and adaptive probing offers no advantage. Figure 2 illustrates the dominance condition under three NN distance distributions: the gap between fixed and QASP curves at any recall target represents data access savings, largest for heavy-tailed distributions at high recall and vanishing for concentrated distributions which is a theoretical worst case scenario. Note that we do not model εL and $\bar{\eta}$ as functions of the recall target, which in practice influences predictor performance.

V. Related Work

QASP targets partitioning-based indexing, the most comprehensive class of vector search methods that partition the

space for pruning and scale to distributed architectures. These include clustering and IVF (Inverted File Index) [5], [9]–[12], multi-dimensional space trees [34]–[37], and vector quantization [10], [11], [38] which uses clustering at its core. Partitioning-based indices achieve sublinear scaling [8] and are well suited to large vector databases because of their parallelizability and resource efficiency. Partitioning also underpins distributed search architectures to distribute vectors across storage units or processors [13]–[16].

Vector search libraries predominantly relied on static empirical parameter settings and target average recall without adapting to query difficulty. These include practices such as defaulting to nprobe = 8 (number of clusters to search), recommending proportional scaling with nlist (total number of clusters) and experimental tuning [39]; setting nprobe to cover 5–10% of the dataset [1]; setting nlist based on dataset size ($4{\cdot}\sqrt{N}$ to $16{\cdot}\sqrt{N}$) with limited nprobe guidance [2] and testing multiple values (1, 4, 16, 64, 256) [40]. SPANN [12] and SQUASH [16] use thresholds to decide which clusters to search.

Popular graph-based methods [41] constrain the candidate list size at query time, traditionally fixed for all queries. Recent methods [4], [20], [42] set this parameter per query through either statistical scoring or reactive early termination. Statistical scoring offers lightweight runtime adaptation but requires distributional assumptions that may not hold across diverse workloads. Reactive early termination provides flexible stopping criteria but periodically invokes models during traversal and incurs per-step inference overhead that constrains practical model size and complexity. These methods target graph indices. For IVF indices, PCE-Net [43] predicts a single nprobe per query, which couples the prediction to a specific recall target and results in higher variance and data access (Section VI-D). QASP instead predicts fine-grained recall contributions over partitions proactively with a single upfront inference. This enables the use of richer model architectures and supports generalization across recall targets.

VI. Performance Evaluation

A. Experimental Setup

All experiments are performed on *Intel(R) Xeon(R) Platinum 8175M CPU @ 2.50GHz & 96 CPU cores*. We use Python 3.10 with Pytorch 1.13 for deep-learning and scikit-learn 1.2.2 for gradient boosting, regression and kmeans++.

1) Vector Datasets: We build QASP on seven datasets from a range of vector spaces, distance measures and application domains. SIFT1M [44], 1 million 128-dimensional SIFT image descriptors; MNIST [45], 60k 784-dimensional vectors trained on handwritten digits; GIST1M [46], 1 million 960-dimensional global color GIST descriptors. DEEP1B_10M, 10 million subset of DEEP1B [47] with 96-dimensional deep learning embeddings; GLOVE-200 [48], 1.2 million 200-dimensional word embeddings trained on Wikipedia; COCO-I2I [49], 113k 512-dimensional vectors for image-to-image retrieval, COCO-T2I [49], 113k 512-dimensional vectors for text-to-image retrieval. Each dataset has a query set with ground-truth $k = 100$ nearest neighbors from exact search; we report $k = 100$ throughout unless stated otherwise.

2) Training, Validation and Test Sets: We use a random sample of $\min(1000, \lfloor |Query\,Set|/10 \rfloor)$ queries for generating offline training and validation sets. Model features, as discussed in section III, are obtained per query and centroid rank by searching up to $\min(200, \lfloor 0.5L \rfloor)$ probes. Ground truth is used to obtain recall target labels. The offline dataset is split 80:20 for training and validation. For testing, 1000 queries are sampled from the remaining query set and features are obtained on-the-fly during search. All reported results use a fixed seed (1234); the reactive policy has minor run-to-run variance from parallel search. See Section VII for model performance and fit analysis.

3) Model Details: Deep Learning (DL) neural architecture of QASP, called QASP-DL, contains 3 hidden layers with hidden dimension of 18 with ReLU activation [50]. Each numerical input feature is mapped to a 4-dimensional embedding via a per-feature learnable linear (affine) layer followed by a ReLU nonlinearity, following the numerical-feature embedding framework for tabular deep learning of [18]. We use batch normalization on input embeddings and dropout with rate 0.1 for regularization. Each DL model is trained for 100 epochs with a constant learning rate of 5e-3 and batch size of 512, though convergence is observed much earlier. The best model is selected using held-out validation set performance. Gradient boosting decision tree (GBDT) model variant of QASP, called QASP-GBDT, is trained using 100 trees with a maximum depth of 3 and a learning rate of 0.1. Polynomial variant of QASP, called QASP-LITE, is a lightweight and easy to adopt alternative in production environment, is fitted using Lasso regression with α=1e-5 on sklearn `PolynomialFeatures` processing with degree 3.

Latency and Throughput - The DL model has a modest parameter count of 1513 trainable parameters and a total memory footprint of just 6-7 KB, making it practical for deployment even in memory-constrained environments. For DL model, mean latency on CPU with batch size 32 is 0.26 ms/batch, throughput 124K queries/sec and P99 latency 0.33 ms using 4 threads. For GBDT, mean latency on CPU with batch size 32 is 0.199 ms/batch, throughput 161K queries/sec and P99 latency 0.55 ms. For QASP-LITE, mean latency on CPU with batch size 32 is 0.06 ms/batch, throughput 524K queries/sec and P99 latency 0.09 ms. All measurement is over 100 runs with 10 warm-up iterations.

4) Index Details: We first build flat IVF indices with nlist as $scalar \times \sqrt{n}$ where n is the base size and use two $scalar$ values $[1, 4]$ to span across different index configurations per dataset. Distance metrics is either Squared Euclidean or Cosine Distance depending on the dataset and the clustering algorithm used is k-means++ [51]. We also build a hierarchical index using two-level kmeans clustering for scaling experiments to larger datasets of size ($> 10MM$). Each level of the index is clustered using fixed nlist as $\sqrt[3]{n}$ that gives equal sized partitions ideal for disk-based experiments.

TABLE I: Comparison of QASP-DL and Oracle Nprobe policy across three recall targets. Oracle Nprobe achieves recall target by over-reading in the 50% Easy split and under-reading in the 50% Hard split, while QASP optimizes search on both splits. We highlight $\bar{r}$ with large deviation from r^* in red otherwise in teal, while best average QVE metrics are emboldened when $\bar{r} \approx r^*$. These results empirically validate Corollary 15 where $A\%$ improvements are better at higher recall targets.

Dataset	Search Policy	$r^* = 90\%$					$r^* = 95\%$					$r^* = 99\%$				
		$\bar{r}$	$\sigma^2 \downarrow$	$\delta \downarrow$	$S\% \uparrow$	$A\% \downarrow$	$\bar{r}$	$\sigma^2 \downarrow$	$\delta \downarrow$	$S\% \uparrow$	$A\% \downarrow$	$\bar{r}$	$\sigma^2 \downarrow$	$\delta \downarrow$	$S\% \uparrow$	$A\% \downarrow$
AVERAGE	Oracle Nprobe	89.84	141.39	9.45	72.19	**3.88**	95.04	49.58	5.25	80.80	7.06	98.43	9.93	1.67	92.18	14.81
	QASP-DL	90.87	**61.95**	**6.30**	**80.03**	3.96	95.09	**28.45**	**3.87**	**85.60**	**5.96**	98.46	**7.18**	**1.36**	**94.56**	**11.47**
50% EASY	Oracle Nprobe	96.47	28.51	8.02	93.41	4.08	98.57	7.45	4.26	97.24	7.34	99.59	1.31	0.89	99.46	15.24
	QASP-DL	91.31	74.95	6.98	80.92	1.85	95.30	36.46	4.24	86.08	2.80	98.53	10.26	1.42	94.71	6.44
50% HARD	Oracle Nprobe	83.17	163.25	10.87	50.97	3.69	91.50	66.01	6.24	64.36	6.78	97.26	13.13	2.45	84.89	14.39
	QASP-DL	90.44	48.17	5.63	79.14	6.07	94.90	20.28	3.51	85.11	9.12	98.41	3.92	1.31	94.40	16.50

B. Query Variability-Aware Evaluation

The traditional method for evaluating vector similarity search results is to measure the deviation of observed average recall from recall target. Given observed average recall, $\bar{r} = \frac{1}{|Q|}\sum_{q\in Q} r(q)$, where $r(q)$ is the observed recall for query q and recall target r^*, traditional evaluation concerns only with $|\bar{r}-r^*|$. While useful as a summary statistic, it masks performance variations across queries and fails to capture user experience consistency. To address this shortcoming, we propose a comprehensive Query Variability-aware Evaluation (QVE) framework with the following metrics to compare query variability even when $\bar{r} \approx r^*$ between search policies.

- **Recall Variance**: $\sigma^2 = \frac{1}{|Q|}\sum_{q\in Q}(r(q) - \bar{r})^2$ - Measures variance of the observed recall across test queries.
- **Absolute Query Deviation**: $\delta = \frac{1}{|Q|}\sum_{q\in Q}|r(q) - r^*|$ - Measures mean of absolute deviation from recall target.
- **Query Satisfaction Rate**: $S\% = \frac{|\{q\in Q: r(q)\geq r^*(1-\gamma)\}|}{|Q|}$ - % of queries satisfied by the search policy, i.e., achieve at least recall r^*. Since hitting r^* exactly is rarely practical, we relax the threshold by a small margin $\gamma = 0.05$.
- **% Data Accessed**: $A\% = \frac{100}{|Q\times D|}\sum_{q\in Q}|\{v \in D : d(v,q) \text{ is calculated}\}|$, % of data accessed.

C. QASP versus Oracle Fixed Policy

We first compare QASP to policies that search a fixed number of probes, which are industry standard where several open source libraries like Faiss [2], Milvus [39], and OpenSearch [3] recommend tuning `nprobe` parameter for IVF for a given dataset and query workload. To construct the strongest such baseline, we fit Oracle Nprobe directly against the target: on the same training queries used for QASP, we sweep `nprobe` and select the smallest value whose mean recall reaches r^*. Because this is the best *fixed* probe count achievable on QASP's own training distribution, Oracle Nprobe is a theoretical upper bound on what any non-adaptive policy can achieve, making it a strong baseline to QASP. See Table I, which contains comparison results averaged across the seven datasets.

Both Oracle Nprobe and QASP achieve similar $\bar{r}$'s close to the target. However, QASP outperforms Oracle Nprobe on QVE metrics achieving 56.2% lower σ^2 and 33.3% lower δ, while satisfying 10.9% more queries and accessing almost the same data (+2.1%) for $r^* = 90\%$. As r^* increases, QASP accesses much fewer data than Oracle Nprobe to achieve the same recall with comparable satisfaction rate. To further understand this behavior, we split the test dataset into two halves — *50% EASY* and *50% HARD*, using Local Relative Contrast feature. We observe that Oracle Nprobe consistently over-reads in the *50% EASY* set and under-reads in the *50% HARD* set. QASP, being query adaptive, does the opposite by reading less in the *50% EASY* set and more in the *50% HARD* set as intended.

D. Baselines

We extend our comparison to proactive baselines from popular vector search libraries and prior work. Several libraries recommend tuning `nprobe` parameter for IVF [2], [3], [39] for which we use Oracle Nprobe. LanceDB [1] suggests searching for a fixed % of vectors for which we obtain Oracle Access % from training dataset. SPANN [12] is a popular clustering index with query-aware pruning that searches until cluster rank j such that $Dist(q, c_j) \leq (1+\epsilon) \times Dist(q, c_1)$. SQUASH [16] also involves a scalar multiple on distance to create a search mask. We obtain Oracle Distance Multiplier to represent these methods. Vexless [14] uses a threshold on absolute distance, for which we derive Oracle Distance. PCE-Net [43] is a learned baseline that predicts a single nprobe per query using neural encoders over the query vector and centroid distance distribution; we evaluate them on the same IVF index for a fair comparison of the probing approaches.

Table II contains comparative results across the seven datasets. First, we observe that all the oracle baselines can be tuned to achieve $\bar{r} \approx r^*$ except Oracle Distance [14]. PCE-Net achieves high recall but at a steep cost: it reads 14.47% of data—more than twice QASP-DL's 5.96%—as its asymmetric loss incentivizes over-probing. Its variance ($\sigma^2 = 73.13$) and deviation ($\delta = 5.49$) also exceed all QASP variants. QASP consistently outperforms all baselines on QVE metrics while achieving recall target, satisfying more queries and accessing fewer vectors. We split test queries into easy and hard. We observe that fixed policies like Oracle Nprobe and Oracle Access% over-read by 4.35% for easy queries and under-read by 2.24% for hard queries. Oracle Distance Multiplier is more adaptive; reading less in easy queries and more in hard queries. However, when compared to QASP models this policy had 114% higher variance. PCE-Net [43], which predicts per

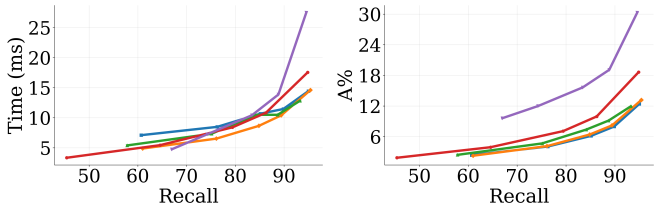
TABLE II: Comparison of QASP to proactive baselines at recall target 95%. QASP outperforms Oracle policies averaged over seven datasets. QASP-DL outperforms other variants achieving least query deviation and data accessed while achieving recall target.

Dataset	Search Policy	$\mathbf{r^* = 95\%}$				
		$\bar{r}$	$\sigma^2 \downarrow$	$\delta \downarrow$	$S\% \uparrow$	$A\% \downarrow$
AVERAGE	Oracle Nprobe [2], [3], [39]	95.04	49.58	5.25	80.80	7.06
	Oracle Access % [1]	95.04	48.46	5.20	80.81	6.97
	Oracle Distance Multiplier [12], [16]	95.36	60.76	5.24	85.62	15.65
	Oracle Distance [14]	93.04	287.93	8.76	84.08	18.62
	PCE-Net [43]	95.23	73.13	5.49	84.54	14.47
50% EASY	Oracle Nprobe [2], [3], [39]	98.57	7.45	4.26	97.24	7.34
	Oracle Access % [1]	98.34	9.22	4.25	95.97	6.96
	Oracle Distance Multiplier [12], [16]	93.66	87.28	6.12	79.08	8.31
	Oracle Distance [14]	98.79	26.58	4.94	97.43	20.59
	PCE-Net [43]	97.84	41.85	4.75	95.70	14.07
50% HARD	Oracle Nprobe [2], [3], [39]	91.50	66.01	6.24	64.36	6.78
	Oracle Access % [1]	91.73	65.17	6.15	65.65	6.98
	Oracle Distance Multiplier [12], [16]	97.06	25.37	4.36	92.16	23.00
	Oracle Distance [14]	87.30	463.19	12.58	70.73	16.65
	PCE-Net [43]	92.63	85.52	6.23	73.39	14.88
AVERAGE	QASP-DL	95.09	28.45	**3.87**	85.60	**5.96**
	QASP-GBDT	95.13	**26.72**	3.92	84.17	6.14
	QASP-LITE	95.40	38.13	4.18	**86.56**	7.13
50% EASY	QASP-DL	95.30	36.46	4.24	86.08	2.80
	QASP-GBDT	95.60	28.29	4.11	85.88	2.65
	QASP-LITE	95.00	47.72	4.69	83.74	3.10
50% HARD	QASP-DL	94.90	20.28	3.51	85.11	9.12
	QASP-GBDT	94.66	23.99	3.73	82.45	9.64
	QASP-LITE	95.77	28.03	3.66	89.39	11.17

query nprobe, exhibits a similar pattern: it over-reads on hard queries ($A\% = 14.88\%$) and under-reads on easy queries ($A\% = 14.07\%$). PCE-Net achieves recall target at higher σ^2 and δ compared to fixed policies like Oracle Nprobe and Oracle Access %. Moreover, while it achieves a higher recall than QASP-DL, it reads more data (14.47% versus 5.96%) and satisfies fewer queries (84.54% versus 85.60%).

E. Deployment Efficiency

Figure 3 evaluates QASP's deployment efficiency in both in-memory and disk-based scenarios. We include Oracle Nprobe and PCE-Net as strong baselines for both latency and recall. For in-memory deployment, we measure total runtime (milliseconds) required to satisfy 95% of queries. QASP incurs a single inference overhead plus search time, while Oracle Nprobe only incurs search time. QASP's constant inference overhead becomes proportionally smaller compared to search time at higher recall targets and with larger datasets (see Section VI-H). We observe diminishing recall returns at higher recalls with Oracle Nprobe compared to QASP models, eventually offsetting inference cost.



(a) In-Memory (b) Disk-Based

Fig. 3: Search time (ms) vs. recall (in-memory) and data accessed vs. recall (disk-based). QASP achieves recall target with less search cost than Oracle Nprobe and PCE-Net, with gains further widening at higher recall targets. Legend: QASP-DL, QASP-GBDT, QASP-LITE, Oracle Nprobe, PCE-Net

For disk-based settings, QASP's benefits are even more pronounced as I/O costs dominate search time. The single inference becomes negligible compared to I/O costs, and QASP directly reduces disk operations. QASP variants consistently read 20-40% less data than Oracle Nprobe across all recall levels, with this efficiency gap widening as recall targets increase (Figure 3). This behavior is supported by the theoretical dominance condition illustrated in Figure 2.

F. Domain Adaptation

1) Cross-Dataset Generalization: We evaluate QASP's generalization to unseen datasets using two QVE metrics ($\bar{r}$ and δ) under two transfer settings: zero-shot (direct application) and few-shot (fine-tuning with 1% of target data; see Section II-D2). As shown in Table III, QASP-DL often generalizes well zero-shot, e.g., GIST1M→SIFT1M achieves $\bar{r} = 98.60$ without tuning. Where gaps exist (e.g., SIFT1M→GIST1M), few-shot fine-tuning with minimal data closes them effectively. Training on diverse sources further helps: a model trained on SIFT1M+MNIST combined outperforms individual models on GIST1M in the zero-shot setting.

TABLE III: Cross-dataset generalization of QASP for recall target 95%. QASP-DL models generalize well under zero-shot and achieve recall close to target, while few-shot training bridges the gap on average with fully trained. QASP models can be fine-tuned with just 1% queries from target domain.

Source	**Target**	**Zero-Shot**		**Few-Shot**		**Fully Trained**	
		$\bar{r}$	$\delta \downarrow$	$\bar{r}$	$\delta \downarrow$	$\bar{r}$	$\delta \downarrow$
GIST1M	SIFT1M	98.60	4.01	93.70	4.23	95.47	3.44
MNIST	GIST1M	95.50	3.97	92.00	5.09	92.30	4.56
SIFT1M	GIST1M	26.80	68.22	92.90	4.36	92.30	4.56
SIFT1M + MNIST	GIST1M	93.70	4.57	94.60	3.87	92.30	4.56
COCOI2I	GLOVE200	93.10	5.52	94.90	3.84	95.23	3.46
GLOVE200	COCOI2I	91.80	6.30	96.80	3.63	95.83	3.98
AVERAGE	-	**83.25**	**15.76**	**94.15**	**4.17**	**93.91**	**4.09**

2) Cross-Index Configuration Generalization: We investigate the robustness of QASP models to varying index configurations by testing how well they perform on the same dataset but with a different number of clusters L. As observed in Table IV, QASP-DL models generalize well on unseen

TABLE IV: Cross-index generalization of QASP on QVE metrics for recall target = 95%. QASP models trained on one index configuration (source) generalize well zero-shot to unseen configurations (target) reducing the need of fine-tuning.

Dataset	**Source Config**	**Target Config**	**Zero-Shot**		**Fully Trained**	
			$\bar{r}$	$\delta \downarrow$	$\bar{r}$	$\delta \downarrow$
SIFT1M	IVF(L=1000)	IVF(L=4000)	85.00	10.83	93.50	4.08
SIFT1M	IVF(L=4000)	IVF(L=1000)	96.30	3.30	95.47	3.44
GIST1M	IVF(L=1000)	IVF(L=4000)	91.50	5.17	91.40	5.17
GIST1M	IVF(L=4000)	IVF(L=1000)	92.30	4.56	92.30	4.56
COCOI2I	IVF(L=336)	IVF(L=1346)	87.80	9.28	95.10	4.08
COCOI2I	IVF(L=1346)	IVF(L=336)	97.20	3.74	95.83	3.98
GLOVE200	IVF(L=1087)	IVF(L=4531)	94.70	3.79	94.70	3.79
GLOVE200	IVF(L=4531)	IVF(L=1087)	95.10	3.59	95.23	3.46
AVERAGE	-	-	**93.84**	**4.78**	**94.29**	**4.07**

index configurations without fine-tuning. Domain adaptation methods remain applicable where few-shot learning can bridge the gap. However, zero-shot application of QASP models to unseen indices produces competitive performance on average.

G. Reactive Complement

We evaluate the performance of QASP's reactive complement described in Section II-E, which guides policy behavior at search. See Algorithm 1 for the pseudocode of the reactive complement policy based on observed discovery rates. The implementation parameters include a smoothing factor $\alpha = 0.3$, $L_\delta = 3$, margins $\epsilon_1 = 0.05, \epsilon_2 = 4.0$ and a threshold of 0.35 on δ_i to capture positive and negative rates. For baselines, we use LAET [4] which pauses search at a fixed nprobe and predicts when to stop, and DARTH [20] which predicts recall at regular intervals to terminate the search early. We tune LAET's *multiplier* parameter to attain recall target of 95%. For DARTH, we set the *ipi* and *mpi* parameters per target recall from its training statistics, and retrain DARTH's predictors for $k = 100$ (the released models use $k = 50$). Table V shows results when all mechanisms are applied on IVF index. Both QASP and QASP + Reactive consistently outperform LAET and DARTH on all three datasets with QASP + Reactive further boosting QASP performance.

TABLE V: Comparison of QASP to reactive baselines at recall target 95%. QASP outperforms both baselines and reactive complement improves QVE metrics at higher average recall.

Dataset	Search Policy	$\mathbf{r^* = 95\%}$				
		$\bar{r}$	$\sigma^2 \downarrow$	$\delta \downarrow$	$S\% \uparrow$	$A\% \downarrow$
SIFT1M	LAET [4]	94.90	109.17	6.18	84.70	4.29
	DARTH [20]	94.20	29.45	4.13	79.50	3.19
	QASP-DL	95.40	18.62	3.37	87.90	2.84
	QASP + Reactive	96.90	13.04	3.51	93.30	3.45
GIST1M	LAET [4]	94.80	119.20	5.61	87.22	10.52
	DARTH [20]	91.80	48.33	5.63	64.67	9.61
	QASP-DL	92.30	47.89	4.72	72.67	6.69
	QASP + Reactive	93.50	36.41	4.19	76.44	7.09
DEEP1B_10M	LAET [4]	94.90	43.81	4.81	82.10	0.85
	DARTH [20]	95.80	27.15	4.23	85.80	1.36
	QASP-DL	95.20	27.42	3.80	86.70	0.88
	QASP + Reactive	96.80	17.57	3.78	92.80	1.08
AVERAGE	LAET [4]	94.87	90.73	5.53	84.67	5.22
	DARTH [20]	93.93	34.98	4.66	76.66	4.72
	QASP-DL	94.30	31.31	3.96	82.42	**3.47**
	QASP + Reactive	95.73	**22.34**	**3.83**	**87.51**	3.87

H. Scaling to Hierarchical Index

We evaluate hierarchical scaling (Section II-D3) by applying a QASP-DL model trained on Deep1B-10M to larger subsets of both SIFT1B and Deep1B (10M–100M). For each subset, we construct a two-level index with `nlist1 = nlist2` = $\sqrt[3]{n}$, yielding balanced partitions suitable for disk-based search. We compare against *Oracle Nprobe*, which optimizes `nprobe1` via binary search with `nprobe2=nlist2`. QASP operates level-wise, predicting adaptive `nprobe1` from first-level features and selectively probing second-level partitions based on predicted recall contribution. As shown in Figure 4, this reduces data access by ≥80% at 99% recall consistently across both datasets (SIFT1B: 82–87%, Deep1B: 80–84%), demonstrating that QASP's efficiency gains generalize across data distributions and scale without retraining.

Algorithm 1 Reactive Adaptive Search for a Single Query

Require: Query q, nearest cluster ordering $\mathcal{C}_q$, planned probes P, recall target r^*
Require: Recall estimates $\{\hat{r}_i\}_{i=1}^{|\mathcal{C}_q|}$ from QASP model
Require: Constants: ϵ_1/ϵ_2 (margins), δ (deviation threshold), α (smoothing factor), L_δ (consecutive deviation limit)
Ensure: Top-k results and actual probes used

$P_{\min} \leftarrow \max(1,\ \lfloor P(1-\epsilon_1) \rfloor - L_\delta)$
$P_{\max} \leftarrow \min(|\mathcal{C}_q|,\ \lceil P(1+\epsilon_2) \rceil - L_\delta)$
Initialize smoothed discovery rate $SDR \leftarrow 0$
Initialize counters $c^- \leftarrow 0$, $c^+ \leftarrow 0$
Initialize previous neighbors set $\mathcal{R}_{\text{prev}} \leftarrow \emptyset$
for $p = P_{\min}$ **to** $P_{\max}$ **do**
  $\mathcal{R}_p \leftarrow \text{SEARCHONECLUSTER}(q, \mathcal{C}_q^p, \mathcal{R}_{\text{prev}})$
  if $p = P_{\min}$ **then** ▷ Skip comparison for first probe
    $\mathcal{R}_{\text{prev}} \leftarrow \mathcal{R}_p$
    continue
  end if
  $\text{DR}_p \leftarrow |\mathcal{R}_p \setminus \mathcal{R}_{\text{prev}}|/k$ ▷ True discovery rate
  $\widetilde{\text{DR}}_p \leftarrow \max(1e^{-3},\ \hat{r}_p - \hat{r}_{p-1})$ ▷ Estimated discovery rate
  $SDR \leftarrow \alpha \cdot \text{DR}_p + (1-\alpha) \cdot SDR$
  $\Delta_p \leftarrow (SDR - \widetilde{\text{DR}}_p)/\widetilde{\text{DR}}_p$
  if $\Delta_p < -\delta$ **then**
    $c^- \leftarrow c^- + 1, \quad c^+ \leftarrow 0$ ▷ Negative discovery
  else if $\Delta_p > \delta$ **then**
    $c^+ \leftarrow c^+ + 1, \quad c^- \leftarrow 0$ ▷ Positive discovery
  else
    $c^-, c^+ \leftarrow 0$
  end if
  if $c^- \geq L_\delta$ **and** $\hat{r}_p \geq r * (1-\epsilon)$ **then**
    break ▷ Over-search detected
  end if
  if $p = P$ **and** $c^- = 0$ **and** $c^+ = 0$ **then**
    break ▷ Consistent with estimation
  end if
  $\mathcal{R}_{\text{prev}} \leftarrow \mathcal{R}_p$
end for
return $\mathcal{R}_p$

VII. Regression Fit Analysis

A. Training Error Analysis

Figure 5 presents training error curves on the seven datasets. All models converge rapidly, with most error reduction occurring within the first 20–30 epochs. Training and validation curves follow similar trajectories with minimal gaps, indicating good generalization without overfitting. Dataset-specific differences in convergence speed and final error levels reflect the varying complexity of the recall prediction task. Overall, QASP trains efficiently across diverse datasets, enabling rapid

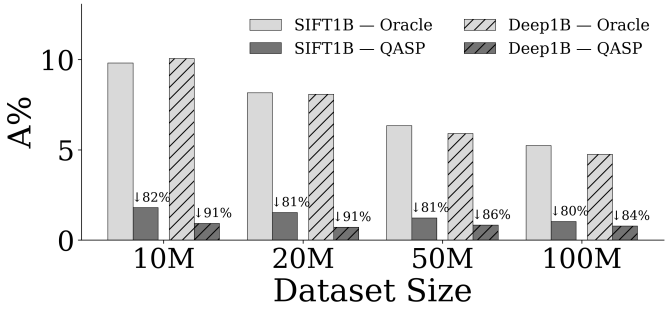
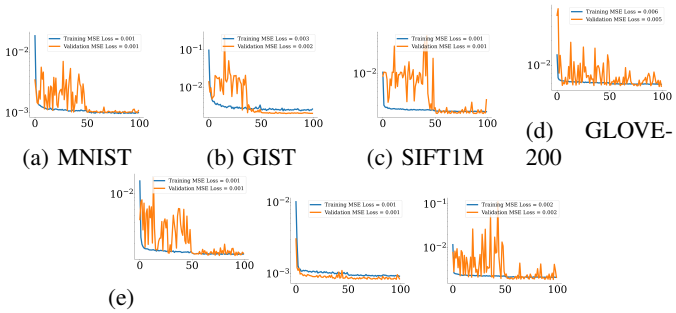


Fig. 4: $A\%$ for SIFT1B and Deep1B subsets (10M–100M) on a two-level hierarchical index at 99% recall target. QASP is trained on a 10M subset and applied at inference without retraining. QASP consistently yields ≥80% reduction in $A\%$.



(a) MNIST (b) GIST (c) SIFT1M (d) GLOVE-200

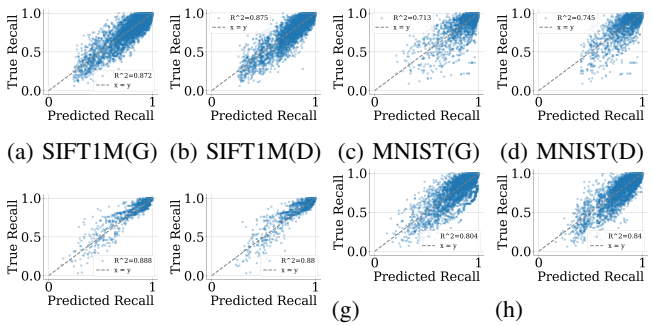
(e) DEEP1B_10M (f) COCOI2I (g) COCOT2I

Fig. 5: Training loss (MSE) over epochs

adaptation to new index configurations and query distributions in practice.

B. Prediction Error Analysis

Figure 6 shows actual versus predicted recall for DL and GBDT models (we omit Lite for brevity). Both models achieve MSE of 0.002 and similar R^2 with DL slightly better. Errors are balanced around the diagonal with a slight bias toward over-prediction—preferable for meeting recall targets. Prediction accuracy remains high across the entire recall range.



(a) SIFT1M(G) (b) SIFT1M(D) (c) MNIST(G) (d) MNIST(D)

(e) GIST(G) (f) GIST(D) (g) DEEP10M(G) (h) DEEP10M(D)

Fig. 6: Prediction error analysis showing actual vs. predicted recall between GBDT (G) and DL (D) recall predictor models.

TABLE VI: Comparison of model fit across all feature ablations for Euclidean and Angular datasets on validation set.

Feature Set	Euclidean Dataset		Angular Dataset	
	MSE ↓	R^2 ↑	MSE ↓	R^2 ↑
ALL	**0.00186**	**0.87123**	**0.00409**	**0.82741**
ALL \ Cumulative Cluster Size	0.00193	0.86634	0.00417	0.82387
ALL \ Relative Distance	0.00246	0.82950	0.00454	0.80837
ALL \ Local Relative Contrast	0.00188	0.86954	0.00432	0.81743
ALL \ Cluster Coeff. of Variance	0.00192	0.86678	0.00413	0.82565
ALL \ Δ(Relative Distance)	0.00191	0.86783	0.00419	0.82296
ALL \ Data Size	0.00188	0.86990	0.00415	0.82464
ALL \ Δ(Cumulative Cluster Size)	0.00186	0.87096	0.00410	0.82686
ALL \ Rank	0.00186	0.87094	0.00416	0.82444
ALL \ Maximum Cluster Distance	0.00187	0.87078	0.00416	0.82407

C. Feature Studies

We fix a set of nine features (Fig.7) and analyze their contributions using SHAP values [52] from the QASP-GBDT model. Cumulative cluster size and relative distance contribute most to positive recall predictions for both Euclidean and Angular datasets, though with high variance—expected since large cluster sizes and relative distances do not guarantee high recall for extremely hard queries. Feature ablation results (Table VI) confirm that cumulative cluster size, relative distance, and local relative contrast are important across both dataset categories, while features such as jump in relative distance are more influential for angular datasets.

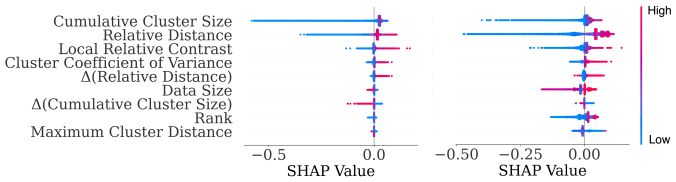


Fig. 7: Feature importance analysis. Left: Euclidean datasets. Right: Angular datasets.

VIII. Conclusion

We introduced QASP to optimize vector search by predicting the complete recall progression curve per query via a single proactive inference, from which a search policy is derived for any recall target. QASP decouples the policy from specific targets or index configurations and enables domain adaptation with zero-shot or minimal fine-tuning. We provide theoretical guarantees, including that a finite sample suffices for convergence independent of dataset size and dimensionality, and a dominance condition where QASP's data access savings over fixed policies grow exponentially in intrinsic dimensionality. Our query variability-aware evaluation demonstrates the importance of minimizing recall variance across queries. QASP achieves significantly lower variance, lower deviation from target, and higher satisfaction rate while accessing similar or less data, with improvements most pronounced for hard queries and high recall regimes. QASP extends to hierarchical partitioning using inference-time scaling alone, achieving 99% recall with 80% less data access. QASP's progressive recall predictions further enable a lightweight reactive complement without additional inference.

References

[1] LanceDB, "IVF-PQ index," LanceDB Documentation, https://lancedb.github.io/lancedb/concepts/index_ivfpq/#query-the-index.

[2] Facebook AI Research, "Guidelines to choose an index," FAISS Wiki, https://github.com/facebookresearch/faiss/wiki/Guidelines-to-choose-an-index.

[3] OpenSearch, "Choose the k-NN algorithm for your billion-scale use case with OpenSearch," Amazon OpenSearch Service Blog, 2022, https://aws.amazon.com/blogs/big-data/choose-the-k-nn-algorithm-for-your-billion-scale-use-case-with-opensearch/.

[4] C. Li, M. Zhang, D. G. Andersen, and Y. He, "Improving approximate nearest neighbor search through learned adaptive early termination," in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020, pp. 2539–2554.

[5] J. Mohoney, D. Sarda, M. Tang, S. R. Chowdhury, A. Pacaci, I. F. Ilyas, T. Rekatsinas, and S. Venkataraman, "Quake: Adaptive indexing for vector search," 2025.

[6] Y. Fu, C. Chen, Y. Chen, W.-F. Wong, and B. He, "Vista: Vector indexing and search for large-scale imbalanced datasets," in *2025 IEEE 41st International Conference on Data Engineering (ICDE)*, 2025, pp. 543–556.

[7] H. Wang, W. Wu, C. Luo, A. Bian, C. Meng, Y. Wu, and J. Sun, "Boosting accuracy and efficiency for vector retrieval with local scaling graph," in *2025 IEEE 41st International Conference on Data Engineering (ICDE)*, 2025, pp. 336–348.

[8] P. Sun, F. Chern, Y. Akhremtsev, R. Guo, D. Simcha, and S. Kumar, "Scaling laws for nearest neighbor search," in *The 1st Workshop on Vector Databases*, 2025.

[9] H. Ferhatosmanoglu, E. Tuncel, D. Agrawal, and A. El Abbadi, "Approximate nearest neighbor searching in multimedia databases," in *Proceedings 17th International Conference on Data Engineering (ICDE)*. IEEE, 2001, pp. 503–511.

[10] E. Tuncel, H. Ferhatosmanoglu, and K. Rose, "Vq-index: An index structure for similarity searching in multimedia databases," in *Proceedings of the 10th ACM International Conference on Multimedia*, 2002, pp. 543–552.

[11] H. Jegou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 1, pp. 117–128, 2010.

[12] Q. Chen, B. Wang, Y. Guo, Y. Zheng, Y. Li, X. Chang, E. Y. Sun, J. Zhang, X. Li, and X. Zhang, "Spann: Highly-efficient billion-scale approximate nearest neighbor search," in *Advances in Neural Information Processing Systems*, vol. 34, 2021, pp. 10 337–10 349.

[13] M. D. Manohar, Z. Shen, G. Blelloch, L. Dhulipala, Y. Gu, H. V. Simhadri, and Y. Sun, "Parlayann: Scalable and deterministic parallel graph-based approximate nearest neighbor search algorithms," in *Proceedings of the 29th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming*, 2024, pp. 270–285.

[14] Y. Su, Y. Sun, M. Zhang, and J. Wang, "Vexless: A serverless vector data management system using cloud functions," *Proceedings of the ACM on Management of Data*, vol. 2, no. 3, pp. 1–26, 2024.

[15] S. Jayaram Subramanya, F. Devvrit, H. V. Simhadri, R. Krishnawamy, and R. Kadekodi, "Diskann: Fast accurate billion-point nearest neighbor search on a single node," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[16] J. Oakley and H. Ferhatosmanoglu, "SQUASH: Serverless and distributed quantization-based attributed vector similarity search," *arXiv preprint arXiv:2502.01528*, 2025.

[17] F. Pedregosa, F. Bach, and A. Gramfort, "On the consistency of ordinal regression methods," *Journal of Machine Learning Research*, vol. 18, no. 55, pp. 1–35, 2017.

[18] Y. Gorishniy, I. Rubachev, and A. Babenko, "On embeddings for numerical features in tabular deep learning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24 991–25 004, 2022.

[19] R. Shwartz-Ziv and A. Armon, "Tabular data: Deep learning is not all you need," *Information Fusion*, vol. 81, pp. 84–90, 2022.

[20] M. Chatzakis, Y. Papakonstantinou, and T. Palpanas, "Darth: Declarative recall through early termination for approximate nearest neighbor search," *Proc. ACM Manag. Data*, vol. 3, no. 4, Sep. 2025.

[21] Y. Li, N. Wang, J. Shi, J. Liu, and X. Hou, "Revisiting batch normalization for practical domain adaptation," 2016.

[22] M. Aumüller and M. Ceccarello, "The role of local dimensionality measures in benchmarking nearest neighbor search," *Information Systems*, vol. 101, p. 101807, 2021.

[23] R. Rivest, "On self-organizing sequential search heuristics," *Communications of the ACM*, vol. 19, no. 2, pp. 63–67, 1976.

[24] D. Pollard, "Convergence of stochastic processes," *Springer Series in Statistics*, 1984.

[25] ——, *Empirical Processes: Theory and Applications*. Institute of Mathematical Statistics, 1990.

[26] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth, "Learnability and the vapnik-chervonenkis dimension," *Journal of the ACM (JACM)*, vol. 36, no. 4, pp. 929–965, 1989.

[27] R. Gupta and T. Roughgarden, "A pac approach to application-specific algorithm selection," *SIAM Journal on Computing*, vol. 46, no. 3, pp. 992–1017, 2017.

[28] M.-F. Balcan, "Data-driven algorithm design," *Beyond Worst Case Analysis of Algorithms (Tim Roughgarden, ed.)*, 2020.

[29] V. N. Vapnik and A. Y. Chervonenkis, "On the uniform convergence of relative frequencies of events to their probabilities," in *Measures of complexity: festschrift for alexey chervonenkis*. Springer, 2015, pp. 11–30.

[30] M. Anthony and P. L. Bartlett, *Neural network learning: Theoretical foundations*. Cambridge University Press, 2009.

[31] P. L. Bartlett and W. Maass, "Vapnik-chervonenkis dimension of neural nets," *The handbook of brain theory and neural networks*, pp. 1188–1192, 2003.

[32] P. W. Goldberg and M. R. Jerrum, "Bounding the vapnik-chervonenkis dimension of concept classes parameterized by real numbers," *Machine Learning*, vol. 18, no. 2-3, pp. 131–148, 1995.

[33] M. Karpinski and A. Macintyre, "Polynomial bounds for vc dimension of sigmoidal and general pfaffian neural networks," *Journal of Computer and System Sciences*, vol. 54, no. 1, pp. 169–176, 1997.

[34] A. Beygelzimer, S. M. Kakade, and J. Langford, "Cover trees for nearest neighbor," in *Proceedings of the 23rd International Conference on Machine Learning (ICML)*. ACM, 2006, pp. 97–104.

[35] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[36] S. M. Omohundro, "Five balltree construction algorithms," *International Computer Science Institute Berkeley*, 1989.

[37] P. Zezula, P. Savino, G. Amato, and F. Rabitti, "Approximate similarity retrieval with m-trees," *The VLDB Journal*, vol. 7, no. 4, pp. 275–293, 1998.

[38] H. Ferhatosmanoglu, E. Tuncel, D. Agrawal, and A. El Abbadi, "Vector-approximation based indexing for non-uniform high dimensional data sets," *Proceedings of the 9th International Conference on Information and Knowledge Management (CIKM)*, pp. 202–209, 2000.

[39] Milvus, "Configure index parameters," Milvus Documentation v2.3.0, 2023, https://milvus.io/docs/v2.3.0/index.md.

[40] Facebook AI Research, "Autotune example," FAISS Source Code, https://github.com/facebookresearch/faiss/blob/main/tutorial/cpp/4-GPU.cpp.

[41] Y. A. Malkov and D. A. Yashunin, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 4, pp. 824–836, 2018.

[42] C. Zhang and R. J. Miller, "Distribution-aware exploration for adaptive HNSW search," *Proceedings of the ACM on Management of Data*, vol. 4, no. 1, 2026.

[43] B. Zheng, Z. Yue, Q. Hu, X. Yi, X. Luan, C. Xie, X. Zhou, and C. S. Jensen, "Learned probing cardinality estimation for high-dimensional approximate NN search," in *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, 2023, pp. 3209–3221.

[44] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, p. 91–110, Nov. 2004.

[45] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[46] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *International journal of computer vision*, vol. 42, pp. 145–175, 2001.

[47] A. Babenko and V. Lempitsky, "Efficient indexing of billion-scale datasets of deep descriptors," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2055–2063.

[48] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543.
[49] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft coco: Common objects in context," 2015.
[50] A. F. Agarap, "Deep learning using rectified linear units (relu)," 2019.
[51] D. Arthur and S. Vassilvitskii, "k-means++: the advantages of careful seeding," in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '07. USA: Society for Industrial and Applied Mathematics, 2007, p. 1027–1035.
[52] S. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," 2017.

IX. AI Assistance Statement

We used Anthropic's Claude model to assist with implementing the PCE-Net baseline from the original paper, as its code was not publicly available. Claude was also used to refactor the codebase for public release and to refine figure aesthetics and layout. All refactored code was manually verified against our original implementation. Experimental design, theoretical analysis, and scientific contributions are solely the authors' work.