On Efficient Approximate Queries over Machine Learning Models

Dujian Ding  
University of British Columbia  
Vancouver, Canada  
dujian@cs.ubc.ca

Sihem Amer-Yahia  
CNRS, Univ. Grenoble Alpes  
Grenoble, France  
sihem.amer-yahia@cnrs.fr

Laks Lakshmanan  
University of British Columbia  
Vancouver, Canada  
laks@cs.ubc.ca

ABSTRACT
The question of answering queries over ML predictions has been gaining attention in the database community. This question is challenging because finding high quality answers by invoking an oracle such as a human expert or an expensive deep neural network model on every single item in the DB and then applying the query, can be prohibitive. We develop a novel unified framework for approximate query answering by leveraging a proxy to minimize the oracle usage of finding high quality answers for both Precision-Target (PT) and Recall-Target (RT) queries. Our framework uses a judicious combination of invoking the expensive oracle on data samples and applying the cheap proxy on the DB objects. It relies on two assumptions. Under the Proxy Quality assumption, we develop two algorithms: PQA that efficiently finds high quality answers with high probability and no oracle calls, and PQE, a heuristic extension that achieves empirically good performance with a small number of oracle calls. Alternatively, under the Core Set Closure assumption, we develop two algorithms: CSC that efficiently returns high quality answers with high probability and minimal oracle usage, and CSE, which extends it to more general settings. Our extensive experiments on five real-world datasets on both query types, PT and RT, demonstrate that our algorithms outperform the state-of-the-art and achieve high result quality with provable statistical guarantees.

PVLDB Reference Format:
Dujian Ding, Sihem Amer-Yahia, and Laks Lakshmanan. On Efficient Approximate Queries over Machine Learning Models. PVLDB, 16(4): 918 - 931, 2022.
doi:10.14778/3574245.3574273

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/DujianDing/AQUAPRO.

1 INTRODUCTION
Several applications at the frontier of databases (DBs) and machine learning (ML) require support for query processing over ML models. In image retrieval for instance, querying a DB corresponds to finding images whose neural representations are close to an input query image, given a distance measure [29, 30]. Similarly, in the medical domain, a typical query would look for patients whose predicted clinical condition is similar to an input patient (see Figure 1) using a Deep Neural Network (DNN) [28, 45]. A straightforward way of answering these queries is to apply the neural models exhaustively on all objects (e.g., images or patients) in the DB, and then return the objects that satisfy the query. This is prohibitive because applying DNNs and involving human expertise are both expensive. In this paper, we propose an approximate query processing approach with provable guarantees that leverages a cheap proxy for the neural model and uses a judicious combination of invoking the expensive oracle model on data samples and applying the cheap proxy on the DB.

The main focus of query processing over ML models has been to ensure efficiency without compromising accuracy [53]. One line of work, query inference, provides native relational support for ML operators using containerized solutions such as Amazon Aurora [17], or in-application solutions such as Google’s BigQuery ML [21] and Microsoft’s Raven [31]. Another line develops adaptive predictions for NNs by pruning examples based on their classification in early layers [8]. Our aim is to enable queries in a way that is agnostic to the underlying prediction model. Hence, we develop an in-application approach where queries can be invoked on any ML prediction model.

Recent work [29, 30, 33, 38] proposes to use cheap proxy models that approximate ground truth oracle labels. Proxies are small neural models that either provide a confidence score [30, 38, 51] or distribution [33] for their predicted labels. Probabilistic predicates (PP) [38] and CORE [51] employ light-weight proxies to filter out unpromising objects and empirically improve data reduction rates in query execution plans. Probabilistic Top-K [33] trains proxy models to generate oracle label distribution and delivers approximate Top-K solutions. Recently, in [30], the authors study queries with a minimum precision target (PT) or recall target (RT), and a fixed user-specified budget on the number of oracle calls. However, (i) setting an oracle budget is hard to get right. An underestimated budget may lead to trivial answers while overestimation causes unnecessary oracle usage, and (ii) setting only a minimum precision or only a minimum recall target, runs the risk of returning valid but uninformative answers: for RT, returning all objects in the DB...
is valid but has very poor precision; for PT, returning the empty set is valid but it has zero recall and is hence not useful in practice.

In this work, we consider oracle/proxy models with multi dimensional outputs. We propose a more useful problem of minimizing oracle usage for finding answers that meet a precision or recall target with provable statistical guarantees while achieving a maximal complementary rate (CR). The CR for an RT (resp. PT) query is precision (resp. recall). More formally, given a PT (resp. RT) query, we seek answers that (1) satisfy a target precision (resp. recall) with a probability higher than a desired threshold and (2) incur a minimal number of oracle calls, and (3) achieve the maximal CR subject to the oracle usage incurred in (2). We aim to minimize oracle usage since the oracle is significantly more expensive than the proxy.

Our problem raises three challenges: (1) identify high quality answers with statistical guarantees, (2) design strategies that exactly or approximately minimize oracle usage, and (3) achieve maximal CR subject to (2). We develop a class of strategies that are agnostic to the prediction model and are applicable both to RT and PT queries. The key idea of our approach is to approximate an oracle with a cheaper proxy model [29, 30]. In practice, the proxy could be a smaller and lower latency neural model. We consider a general pipeline for query answering which consists of three stages: (1) apply proxy on the DB, (2) sample & probe with oracle, (3) compute and return answers (see Figure 1). We instantiate our pipeline under two alternative assumptions. Under the Proxy Quality assumption, the proxy quality w.r.t. the oracle is quantified in a probabilistic manner which allows us to return high quality answers right after applying the proxy on the DB. We develop Algorithm PQA which efficiently finds high probability valid answers of maximal expected CR with zero oracle calls. We additionally design Algorithm PQE, a heuristic extension to PQA, to calibrate the correlation between the oracle and the proxy by incurring some oracle calls. If the proxy quality is hard to quantify, we have the Core Set Closure assumption under which we uniformly sample and probe a subset of objects to estimate valid answers to a given query. We introduce the notion of core set to find the optimal sample size and number of samples so as to ensure a minimal expected oracle usage to identify a valid answer with high probability. We use the proxy to improve answer CR heuristically. This leads to Algorithm CSC, which efficiently returns high probability valid answers with a minimal expected number of oracle calls, and an empirically good CR. We also design Algorithm CSE, a generalization of CSC, which calibrates core sets with extra oracle calls and ensures high success probability.

We conduct experiments on five real-world datasets and compare our algorithms to four baselines from recent work: (1) SUPG [30], (2) Top-K [33], a probabilistic Top-K approach that uses oracle score distribution to deliver approximate Top-K answers, (3) Sample2Test, a sample-based baseline adapted from the literature [38], and (4) Scan2Test, a simple baseline that returns answers by applying oracle on all objects, which is also compared with in [33]. Our experiments demonstrate that our algorithms find high quality answers with statistical guarantees even when baselines fail. More specifically, we analyze PQA and verify the optimality of its CR and success probability guarantee under the Proxy Quality assumption. We compare PQA with Top-K on a synthetic dataset and demonstrate that PQA returns high quality answers with zero oracle call while Top-K incurs a huge oracle cost. We analyze CSC to demonstrate its minimal oracle usage and success probability guarantee under the Core Set Closure assumption. We compare PQE, CSE, and baselines in terms of success probability and CR, under various oracle settings. We find that for RT queries, CSE has the best oracle efficiency and for PT queries, PQE is the most oracle efficient approach. Finally, we study scalability and find that CSE is the most efficient approach outperforming the strongest baseline by up to 87.5%.

In sum, we make the following contributions.

- We propose the problem of answering PT and RT queries with minimal oracle usage and maximal CR while meeting precision or recall targets with high probability (§ 2).
- We propose two assumptions (Proxy Quality and Core Set Closure), around which we develop four algorithms (PQA, PQE, CSC, and CSE) to solve the problem efficiently (§ 4).
- We run extensive experiments on five real-world datasets (§ 5) and show that: (i) our approaches yield valid answers with high probability; (ii) our approaches significantly outperform the state of the art w.r.t. CR and cost.

Complete details of proofs as well as additional experiments can be found in the full version [19].

2 PROBLEM STUDIED

2.1 Use Cases

Example 1 (Image Retrieval). The problem is to find images similar to a query image [13, 18, 52]. Metadata-based approaches use textual descriptions of images for quickly measuring similarity, but their quality heavily relies on image annotations [18].

Example 2 (Preventive Medicine). One of the greatest obstacles of preventive medicine is the limited time a physician has [9, 37, 45, 49]. Clinical Risk Prediction Models (CRPMs) are being developed to facilitate decision-making. CRPMs serve as prognosis prediction systems and predict the occurrence of specific diseases based on personalized medical records. Our goal is to extend queries to include CRPMs while offering statistical guarantees [15, 35, 46].

Example 3 (Video Analytics). While DNNs have become effective for querying videos [43], their inference cost becomes prohibitive as the model size increases. For example, to identify frames with a given class (e.g., ambulance) on a month-long traffic video, an advanced object detector such as YOLOv2 [44] needs about 190 GPU hours and $380 for a cloud service [26]. A specialized model can achieve high efficiency, e.g., up to 340x faster than the full DNN, with sacrificed accuracy [29]. Our goal is to efficiently generate high quality query answers by balancing the use of expensive high-accuracy models and cheap low-accuracy proxies [26, 29, 30].

2.2 Query

Our queries generalize Fixed-Radius Near Neighbor (FRNN) queries [5]. Given a dataset D, a query object q, a radius r, and a distance
function \( dist \), an FRNN query asks for all near neighbors of \( q \) within radius \( r \), i.e., \( NN(q,r) = \{ x \in D \mid dist(x,q) \leq r \} \). In this paper, we are mainly interested in near neighbors of objects w.r.t. latent features, using a distance function defined on these features. In preventive medicine [45, 49], a latent feature may be the infection risk of a disease, which can be inferred from patient history, drug usage, and demographics. Latent features can be discovered by human experts [38] or powerful neural models, which we refer to as oracle, denoted \( O \). The near neighbors of query object \( q \) w.r.t. \( O \) and radius \( r \) are defined as \( NN^O(q,r) = \{ x \in D \mid dist(O(x),O(q)) \leq r \} \). We will use the notation \( NN^O \) when the query object \( q \) and radius \( r \) are clear from the context. An object \( x \in D \) is an oracle neighbor of a query object w.r.t. radius \( r \) if \( x \in NN^O \). Retrieving the exact \( NN^O \) requires calling the oracle on every single object in the \( DB \), which is prohibitively expensive. Instead, we are interested in finding high quality answers with high probability (w.h.p.).

For any subset \( S \subseteq D \), we denote by \( N_S = |S \cap NN^O| \) the number of oracle neighbors in \( S \). Define:

\[
M_p(S) = N_S/|S| \quad M_v(S) = N_S/|NN^O| \tag{1}
\]

A query specifies a user-given measure \( M \), which can be either \( M_p \) for precision or \( M_v \) for recall, to measure answer quality, and a target \( y \in (0,1) \). In the former case, it is called a Precision-Target (PT) query and in the latter, Recall-Target (RT) query. We call \( Ans \subseteq D \) a valid answer iff \( M(Ans) \geq y \). For \( M \), we use \( \mathbb{M} \) to denote its complementary rate (CR): when \( M = M_p \) (resp., \( M_v \)), \( \mathbb{M} \) stands for \( M_r \) (resp., \( M_p \)). Given a query, we are interested in returning valid answers w.h.p. For any \( S \subseteq D \), the probability of success for \( M(S) \) is \( PoS(S,M,y) := \Pr[M(S) \geq y] \). We generalize FRNN queries to Approximate Oracle-Sensitive FRNN (AOS-FRNN) queries.

Definition 2.1 (AOS-FRNN Query). Given a dataset \( D \), a query object \( q \), a radius \( r \), a failure rate \( \delta \), a main measure \( M \) and corresponding target \( y \in (0,1) \), an AOS-FRNN query asks for a valid answer \( Ans \subseteq D \) w.h.p., i.e., such that \( PoS(Ans,M,y) \geq 1 - \delta \).

Effectively processing an AOS-FRNN query requires determining: (1) How many oracle calls are required to find a valid answer w.h.p.? and (2) How good is the returned answer under a given CR? The first question is important since oracle invocations are expensive and must be reduced. The second question is important because a technically valid answer could be uninformative. For instance, if a user specifies \( M = M_p \) with a high target \( y \), the empty set is always a valid answer. Similarly, returning (nearly) the whole dataset is always a valid answer when \( M = M_v \).

Problem 1 (AOS-FRNN Problem). Given a dataset \( D \) and an AOS-FRNN query \( Q \), find a valid answer \( Ans \subseteq D \) to \( Q \) w.h.p. such that (i) the number of oracle calls incurred is minimal and (ii) the complementary rate \( \mathbb{M}(Ans) \) achieved is maximal subject to (i).

The AOS-FRNN Problem is challenging given that we want to optimize two objectives (i.e., oracle usage and CR) under validity and success probability constraints. We will show that under certain conditions, we can efficiently return high probability valid answers with minimal or zero oracle calls and maximal expected CR.

3 APPROACH OVERVIEW

The key idea of our approach is to approximate an oracle with a cheaper proxy model [29, 30]. In practice, compared to an expensive oracle \( O \), a proxy \( P \) could be a smaller and lower latency neural model. For brevity, when a query object \( q \) is clear from the context, we use \( dist^P(x) \) (resp. \( dist^O(x) \)) to denote \( dist(P(x),O(q)) \) (resp. \( dist(O(x),O(q)) \)), for any \( x \in D \).

Given a dataset \( D \), define an index function \( I : D \to \{ i \mid 1 \leq i \leq |D| \} \) that enumerates data objects in increasing order of their proximity distance, i.e., \( \forall x_1,x_2 \in D, I(x_1) \leq I(x_2) \) if \( \text{dist}^P(x_1) \leq \text{dist}^P(x_2) \). Denote by \( D_k = \{ x \in D \mid 1 \leq I(x) \leq k \} \) the \( k \) nearest neighbors of the query object \( q \) w.r.t. the proximity distance. \( D_0 \) is the empty set. Given a query object \( q \), for \( x \in D \), we say that \( k \) is the proxy index of \( x \) if \( I(x) = k \). In this case, we call \( D_k \) the proxy prefix of \( x \).

To solve the AOS-FRNN Problem with guarantees, we examine two alternative assumptions:

Assumption 1 (Proxy Quality): When the proxy quality w.r.t. the oracle can be quantified in a probabilistic manner, we aim to find high probability valid answers of maximal expected CRs with no oracle calls. We develop Algorithm PQA to do that. For \( x \in D \), PQA assumes the conditional probability of \( dist^O(x) \), given \( dist^P(x) \). We can show that this assumption holds as long as data is i.i.d. (see § 4.1.1). This allows it to compute the success probability \( PoS(S,M,y) \) and expected CR \( \mathbb{E}[\mathbb{M}(S)] \) for any answer \( S \subseteq D \). We prove that the optimal answer to any given query is \( D_{k^*} \) for some \( 0 \leq k^* \leq |D| \). The optimal answer satisfies validity w.h.p. and has maximal expected CR. As \( k^* \) is not known a priori, we explore the monotonicity of \( PoS(D_k,M,y) \) and \( \mathbb{E}[\mathbb{M}(D_k)] \) w.r.t. \( k \) in order to efficiently identify \( D_{k^*} \). For RT queries, \( PoS(D_k,M,y) \) monotonically increases as \( k \) increases. We use binary search to identify the smallest \( k = k^* \) such that \( PoS(D_k,M,y) \geq 1 - \delta \). Next, we find \( k \leq k^* \leq |D| \) which maximizes \( \mathbb{E}[\mathbb{M}(D_k)] \) and return \( D_{k^*} \) as the answer. For PT queries, \( \mathbb{E}[\mathbb{M}(D_k)] \) monotonically increases as \( k \) increases. Thus, we incrementally compute \( PoS(D_k,M,y) \) for \( 0 \leq k \leq |D| \) and set \( k^* \) as the largest \( k \) s.t. \( PoS(D_k,M,y) \geq 1 - \delta \). We return \( D_{k^*} \) as the answer. It is easy to see that \( D_{k^*} \) is the optimal answer and no oracle call is invoked in computing it.

Example 4. A (synthetic) illustrative example is shown in Figure 2.1. Consider a dataset \( D = \{ x_1,x_2,\ldots,x_k \} \). We show how to use PQA to solve the example RT and PT queries with \( y = 0.9, \delta = 0.1 \) and a ground truth \( NN^O = \{ x_1,x_2,x_3 \} \). We first compute proxy distance \( \text{dist}^P(x_i) \) for each \( x_i \in D \) and derive the oracle distance distribution \( \text{Pr} [ \text{dist}^O(x_i)|\text{dist}^P(x_i)] \) according to our assumption, which allows us to compute \( \text{PoS}(S,M,y) \) and \( \mathbb{E}[\mathbb{M}(S)] \) for any \( S \subseteq D \). We want to efficiently find the optimal answer \( D_{k^*} \). In this example, \( I(x_i) = i \) and \( D_k = \{ x_1,x_2,\ldots,x_k \} \). For the RT query, we use binary search to find \( k = 5 \), i.e., the smallest \( k \) satisfying \( \text{PoS}(D_k,M_p,y = 0.9) \geq 1 - \delta = 0.9 \). Next, we compute expected precision and return \( D_3 \) as the answer since \( \mathbb{E}[M_p(D_3)] = 0.75 \geq \mathbb{E}[M_p(D_k)] \) for any \( k \leq k \). For the PT query, we compute \( \text{PoS}(D_k,M_p,y = 0.9) \) for \( 0 \leq k \leq |D| \) and return \( D_3 \) as the answer since \( k = 3 \) is the largest \( D_k \) satisfying \( \text{PoS}(D_k,M_p,y = 0.9) \geq 1 - \delta = 0.9 \).

1 All numbers are synthetic and are used to illustrate the operational workflow of our algorithms. We provide details of each computational step in § 4.
the RT query, \( x_5 \) is the only oracle neighbor whose proxy prefix is a valid answer. Therefore, \( C = \{ x_5 \} \) and \( C \) is closed. We can derive the optimal values \( s^* = 3 \) and \( m^* = 2 \), and uniformly draw samples \( S1, S2 \) from \( D \). We then apply oracle on each \( x_i \in S = S1 \cup S2 \) and compute dist\(^O\)(\( x_i \)) accordingly. At the end, we set \( k_S = S \) and return \( D_S \) as the answer since \( x_5 \) has the largest proxy index among sampled oracle neighbors \( x_1, x_2, x_3 \). For the PT query, the core set is \( C = \{ x_1, x_2, x_3 \} \), which is closed. Similarly, we first derive the optimal values \( s^* = 2 \) and \( m^* = 2 \), and draw \( S1, S2 \) accordingly. Next, we apply the oracle on samples and compute the corresponding oracle distance. At the end, we set \( k_S = 2 \) and return \( D_S \) as the answer since \( x_5 \) has the smallest proxy index among sampled oracle neighbors \( x_1, x_2, x_3 \).

In case these assumptions do not hold, we develop PQE and CSE. PQE is a heuristic extension to PQA which calibrates oracle distance distribution by incurring some oracle calls. CSE complements CSC and ensures high success probability in general. The workflow and performance of all four approaches are summarized in Figure 4 and Table 1.

### Table 1: Performance of different approaches for queries with specified \( \delta \). Provable guarantees are highlighted. Empirical performance is described by “high”, “small”, and “good”.

| Approach | Success Prob. | Oracle Usage | CR | Assumption |
|----------|---------------|--------------|----|------------|
| PQA      | \( \geq 1 - \delta \) | MIN          | CR | Yes        |
| CSC      | \( \geq 1 - \delta \) | small        | CR | Yes        |
| CSE      | \( \geq 1 - \delta \) | small        | CR | No         |
| PQE      | high          | small        | CR | No         |

### Table 2: Notation Summary

| Symbol | Description |
|--------|-------------|
| dist\(^O\)(\( x \)) | oracle distance |
| dist\(^P\)(\( x \)) | proxy distance |
| \( NNO \) | oracle neighbors in DB |
| \( N_S \) | # of oracle neighbors in \( S \) |
| \( k \) | \( k \)-proxy-nearest neighbors |
| \( M, \overline{M} \) | main/comp. measure |

We will use \( M \) and \( \overline{M} \) when results hold for both \( PT \) and \( RT \). We next describe our algorithms and provide a theoretical analysis.

### 4 FORMAL ANALYSIS AND ALGORITHMS

#### 4.1 Proxy Quality

In § 4.1.1, we formally state the Proxy Quality assumption and show how the success probability of a set \( S \subseteq D \) can be computed. Then, we develop Algorithm PQA based on this assumption (§ 4.1.2)
and analyze answer optimality (§4.1.3). In §4.1.4, we develop Algorithm PQP to extend PQP to more general settings.

4.1.1 Proxy Quality Assumption. In many real-world applications, data is collected in i.i.d. manner [33, 38]. In our problem setting, the oracle and proxy are provided as input and serve as deterministic functions mapping a data object \( x_i \in D \) to its prediction \( O(x_i) \) or \( P(x_i) \). The difference between oracle and proxy distances to a given query object can be seen as i.i.d. random variables, whose i.i.d. property comes from the underlying data collection process. Formally, the assumption states that given a query, the deviations between the proxy and oracle distances of different objects \( x_i \in D \) are i.i.d. random variables: for \( x_i \in D, \epsilon_i = dist^O(x_i) - dist^P(x_i) \), where \( \epsilon_i \) are i.i.d., \( \epsilon_i \sim X \). In Figure 5 we report the distribution of \( \epsilon_i \) on two real-world datasets, Mimic-III [27] and night-street [10]. It is clear that, with high frequency, \( \epsilon_i \) takes values on close to zero, which indicates that the proxy is of good quality and can properly approximate the oracle predictions.

Under this assumption, we can compute the oracle distance distribution for any \( x_i \in D \), after observing the proxy distance. The conditional probability of \( x_i \in D \) being an oracle neighbor is:

\[
Pr[x_i \in NN^O | dist^O(x_i) \leq r | dist^P(x_i)] = Pr[\epsilon_i \leq r | dist^P(x_i)]
\]

The RHS of Eq. 2 is the cdf of \( \epsilon_i \sim X \) evaluated at \( r - dist^P(x_i) \), i.e., \( CDF_X(r - dist^P(x_i)) \). For simplicity, define \( \phi(x_i) := CDF_X(r - dist^P(x_i)) \) and \( \Phi(D) := \{\phi(x_i) | x_i \in D\} \). Notice, \( \phi(x_i) \) provides the probability that \( x_i \) is an oracle neighbor. The overall success probability uses the possible world semantics [40]. The success probability of a subset \( S \subseteq D \) equals the sum of probabilities of all possible worlds in which \( S \) has a high precision or recall w.r.t. the target \( y \). To compute the success probability of \( S \), we seek the likelihood of any \( S \subseteq D \) containing a certain number of oracle neighbors.

Recall that for any subset \( S \subseteq D \), \( NN = |S \cap NN^O| \) is the number of oracle neighbors in \( S \). \( NN \) is thus a random variable equal to the sum of \( |S| \) independent Bernoulli trials, each of which has a success probability \( \phi(x_i) \), \( x_i \in S \). Let \( p_{NN}(k) := Pr[NN = k] \) be the probability mass function for any \( S \subseteq D \) and \( 0 \leq k \leq |S| \). We next discuss how to compute it efficiently.

An important fact is that, given \( S \subseteq D \), \( x_i \notin S \), \( p_{NN}(k) \), and \( p_{NN} \), satisfy the following recurrence relation:

\[
p_{NN_{S \cup \{x_i\}}}(k) = p_{NN}(k - 1) \cdot \phi(x_i) + p_{NN}(k) \cdot (1 - \phi(x_i))
\]

(3) for \( 0 \leq k \leq |S| + 1 \). Eq. 3 says how to compute the probability mass function \( p_{NN_{S \cup \{x_i\}}} \) from \( p_{NN} \), for any \( S \subseteq D \) and \( x_i \notin S \). This recurrence relation directly suggests a way to compute \( p_{NN} \) for any \( S \) with incremental updates, called direct convolution [7]. We start from \( S = \emptyset \) and apply Eq. 3 recursively to compute \( p_{NN} \), for any \( S \subseteq D \). \( p_{NN} \) is implemented by an array (we abbreviate \( \phi(x_i) \) as \( \phi_i \)). We initialize the array \( p_{NN}[0] = 1 \). We then iteratively update \( p_{NN} \) by including \( x_i \in S, 1 \leq i \leq |S| \), in any order. The distribution updates are a direct implementation of Eq. 3.

We now discuss how to use \( p_{NN} \) to compute \( PoS(S, M, y) \), the success probability for \( S \) to be a valid answer. We have the following fact, where \( \bar{S} := D \setminus S \):

**Fact 1.** Given \( S \subseteq D \) and \( y \in (0, 1) \),

\[
PoS(S, M_p, y) = Pr\left[\frac{NN}{|NN^O|} \geq y\right] = \sum_{k=0}^{|S|} p_{NN}(k)
\]

(4)

\[
PoS(S, M_o, y) = Pr\left[\frac{\bar{NN}}{|NN^O|} \geq y\right] = \sum_{j=0}^{|S|} p_{NN}(j) \sum_{k=0}^{|j-y|/y} p_{NN}(k)
\]

(5)

For PT queries, the precision of \( S \subseteq D \) increases as \( S \) contains more oracle neighbors. The probability of \( S \) having a precision no less than \( y \) equals the probability of \( S \) containing at least \( |y| \) oracle neighbors, i.e., \( Pr[NN \geq |y|] \). Eq. 4 gives this probability.

For RT queries, the recall of \( S \subseteq D \) increases as \( S \) contains more oracle neighbors relative to the complement \( \bar{S} = D \setminus S \). If \( S \) contains \( 0 \leq j \leq |S| \) oracle neighbors, the conditional probability of \( S \) having a recall no less than \( y \) equals the probability that \( \bar{S} \) contains no more than \( j(1 - y)/y \) oracle neighbors, i.e., \( Pr[\bar{NN} \leq j(1 - y)/y] \). By the law of total probability [23], the overall success probability \( PoS(S, M_o, y) \) equals the summation of the product between the conditional success probability, \( Pr[NN \leq j(1 - y)/y] \), and the marginal probability, \( Pr[\bar{NN} = j], 0 \leq j \leq |S| \). Using Eq. 5, we use \( p_{NN} \) and \( p_{NN} \) to compute this probability.

Fact 1 gives a direct way to compute \( PoS(S, M, y) \) for any \( S \subseteq D \) for a given query. We also leverage Eq. 4 and Eq. 5 iteratively.

4.1.2 Algorithm PQP. We develop PQP (Algorithm 1) which returns high probability valid answers with zero oracle calls and maximal expected CR, under the Proxy Quality assumption. For PT queries, PQP-PT computes the largest \( k \) s.t. \( PoS(D_k, M_p, y) \geq 1 - \delta \), \( 0 \leq k \leq |D| \), denoted \( k^* \). Notice that \( PoS(S, M_p, y) \) can be derived from \( p_{NN} \), in linear time, and \( p_{NN} \) can be computed from \( p_{NN(x_i)} \) in linear time, for any \( x_i \in S \subseteq D \). PQP-PT incrementally computes \( p_{NN} \) for each \( 0 \leq k \leq |D| \) and \( PoS(D_k, M_p, y) \) accordingly. At the end, PQP-PT returns \( D_{k^*} \) where \( k^* = \max(0 \leq k \leq |D| | PoS(D_k, M_p, y) \geq 1 - \delta) \). For RT queries, PQP-RT uses binary search to identify the smallest \( k = k^* \) such that \( PoS(D_k, M_o, y) \geq 1 - \delta \). Next, PQP-RT computes the expected CR of \( D_k \) for each \( k \leq k \leq |D| \), and returns \( D_{k^*} \) where \( k^* = \max(0 \leq k \leq |D| | PoS(D_k, M_o, y) \geq 1 - \delta) \).

The algorithm is presented in Algorithm 1. PQP-PT is given in lines 1-8. In lines 2-4, we incrementally compute \( p_{NN} \) for \( 0 \leq k \leq |D| \). In lines 6-8, we keep tracking the largest \( k = k^* \), \( 0 \leq k \leq |D| \), such that \( PoS(D_k, M_p, y) \geq 1 - \delta \), and return \( D_{k^*} \) as the answer. The overall time complexity is \( O(|D|^2) \). PQP-RT is given in lines 9-26. In lines 10-18, we use binary search to find the smallest \( k = k^* \) such that \( PoS(D_k, M_o, y) \geq 1 - \delta \). Next, in lines 20-26, we compute \( E[M_p(D_k)] \) for each \( k \leq k \leq |D| \) and return \( D_{k^*} \) with the maximal expected CR. Binary search invokes \( O(|log(|D|)|) \) times \( P_{NN} \) computation, each of which is of \( O(|D|^2) \). The overall time complexity is therefore \( O(|log(|D|)||D|^2) \).

4.1.3 PQP Optimality. We first show that there exists some \( D_{k^*} \), s.t. it is an optimal answer. We then explore the monotonicity relation between \( D_k \) and \( D_{k+1} \) w.r.t. success probability and expected
CR, to efficiently find $D_k$. Finally, we show that answers returned by PQA are optimal for any query (proofs in the full version [19]).

For $S \subseteq D$, we are interested in two operations to generate new answers: (i) replace $x_i$ with $S$ has a new object $x \not\in S$. We first show that for any $S \subseteq D$, both success probability $PoS(S, M, y)$ and expected CR $\mathbb{E}(M(S))$ are monotone under the replacement operation.

**Lemma 1 (Monotonicity of Replacement).** Let $S \subseteq D$, $x_i \in S$, and $x_j \not\in S$. Denote $S' = S \cup \{x_j\} \setminus \{x_i\}$. For all $y \in (0, 1)$, if $\phi(x_i) \leq \phi(x_j)$, then

$$PoS(S, M, y) \leq PoS(S', M, y) \quad \text{and} \quad \mathbb{E}[M(S)] \leq \mathbb{E}[M(S')]$$

**Proof Sketch.** The proof leverages the notion of usual stochastic order [41].

Lemma 1 says that, given $S \subseteq D$, if we replace $x_i \in S$ with $x_j \not\in S$, where $x_j$ is more likely to be an oracle neighbor, both the success probability and the expected CR of $S$ will monotonically increase, for a given query. Lemma 1 can be used to prune out a majority of unpromising solutions in the early stage of query processing. Specifically, given a query, we show that for any $0 \leq k \leq |D|$, $D_k$ is optimal among all answers of size $k$. Recall $D_k$ is the set of $k$ nearest neighbors of the query object w.r.t. the proxy distance. Formally,

**Theorem 4.1.** For all $y \in (0, 1)$, $\forall 0 \leq k \leq |D|$, $D_k$ has the highest success probability and expected CR among all $S \subseteq D$ with $|S| = k$.

Theorem 4.1 entails that, given a query, there exists some $0 \leq k^* \leq |D|$ such that $D_{k^*}$ is guaranteed to be an optimal answer. We study the append operation and have the following result.

**Lemma 2 (Monotonicity of Append).** For all $y \in (0, 1)$ and $0 \leq k \leq |D| - 1$,

$$PoS(D_k, M_r, y) \leq PoS(D_{k+1}, M_r, y) \quad \mathbb{E}[M_r(D_k)] \leq \mathbb{E}[M_r(D_{k+1})]$$

Lemma 2 states that increasing $k$ leads to an increase both in the probability for $D_k$ to have a high recall and its expected recall. In other words, the success probability of $D_k$ monotonically increases for RT queries, and the expected CR of $D_k$ monotonically increases for PT queries, as $k$ increases.

By Theorem 4.1 and Lemma 2, for any given query, the answer $D_k^*$ returned by Algorithm PQA clearly has high success probability and the maximal expected CR, implying it is an optimal answer.

4.1.4 Algorithm PQE. Recall that Algorithm PQA requires $\Phi(D)$ as an input. In a general setting, when $\Phi(D)$ is unknown or proxy Quality Assumption does not hold, we heuristically fit a normal distribution by sampling and probing on a limited number of objects, where the limit is controlled by a budget parameter. The resulting algorithm is PQE (Algorithm 2).

That is, in PQE, we employ $\epsilon_i \sim N(\mu, \sigma)$ for all $x_i \in D$. Specifically, we choose $\mu = 0$, which amounts to assuming that the proxy is an unbiased estimator of the oracle. For $\sigma$, given a budget $b$, we sample and probe $b$ objects to estimate $\sigma$, denoted $\tilde{\sigma}$. We further introduce a hyper-parameter $\sigma_0$ to represent the deviation from the Proxy Quality assumption. In the ideal case where Proxy Quality holds, $\sigma_0 = 0$. We heuristically choose $\sigma = \tilde{\sigma} + \sigma_0$. We use $N(\mu, \sigma)$ to compute $\Phi(D)$ and pass it to PQA to find the answers.

**Algorithm 2: PQE**

1. **Function** PQE($\Phi_D$, $\delta$, $\delta_r$, $\sigma_0$):
2. $S \leftarrow \text{Sample}(D, b)$
3. $\sigma \leftarrow \sigma_0 + \text{std}([\text{dist}^{O}(x) - \text{dist}^{P}(x) \mid x \in S])$
4. $\Phi_D \leftarrow \{CDF_{N(\mu, \sigma)}(r - \text{dist}^{P}(x)) \mid x \in D\}$
5. if RT query then
6. return PQA-RT($\Phi_D$, $\delta$)
7. else
8. return PQA-PT($\Phi_D$, $\delta$)

Algorithm 2 details the steps. In lines 2-4, we draw a sample $S \subseteq D$ of $b$ objects to estimate $\sigma$ and compute $\Phi(D)$. In lines 5-8, we invoke Algorithm PQA with $\Phi(D)$ for PT or RT queries. The overall time complexity is dominated by Algorithm PQA; the additional time complexity on top of PQA is $O(|D|)$.

We now introduce Core Set Closure assumption, our second alternative assumption, and develop two algorithms CSC and CSE.

### 4.2 Core Set Closure

In § 4.2.1, we formally introduce the Core Set Closure assumption and show how to find the optimal sample and probe strategy when core set size is known. We also analyze the case when core set size is unknown and show how to ensure high success probability. In § 4.2.2, we develop Algorithms CSC and CSE based on this. In § 4.2.3, we discuss how to support progressive query processing.
4.2.1 Core Set Closure Assumption. For a query, we define the core set as the set of all oracle neighbors whose proxy prefix is a valid answer. We use $c$ to denote the size of a given core set $C$. A core set $C$ is closed w.r.t. an RT (resp. PT) query if for any $x \in C$, any oracle neighbor whose proxy index is larger (resp. smaller) than that of $x$ is also an element of $C$. Core Set Closure assumption says that, for any given query, the core set is closed w.r.t. that query.

For RT queries, the core set is always closed, because as the proxy index of oracle neighbors increases, the recall of corresponding proxy prefix monotonically increases. For PT queries, with a properly tuned proxy, the core set is likely to be closed in practice. In Figure 6, we report the average Precision($D_k$) over 100 random queries on two real datasets, Mimic-III [27] and night-street [10]. It is clear that the precision of proxy prefix $D_k$ monotonically decreases as $k$ increases on both datasets, which shows the core set closure property for PT queries.

We uniformly draw $m$ samples of size $s$ from $D$ to derive $\kappa_S$ where $S$ is the union of samples, and return $D_{k_S}$ as the answer. Recall that $\kappa_S$ is the largest (resp. smallest) $I(x)$ for RT (resp. PT) queries, where $x$ is determined to be an oracle neighbor by probing $S$ (see §3, Assumption 2). If the core set $C$ is closed w.r.t. a given query, the recall probability of $D_{k_S}$ is the likelihood of $S$ intersecting with $C$, i.e., $Pr(S \cap C = 0)$. Since samples are drawn uniformly, we have $Pr(S \cap C \neq \emptyset) = 1 - ((|D| - c)/|D|)^m = 1 - (\prod_{i=0}^{\lfloor s/\delta \rfloor - 1} (|D| - s - i))^{m}$, where $s$ is the sample size and $m$ is the number of samples. We denote $f(|D|, s, m) = 1 - (\prod_{i=0}^{\lfloor s/\delta \rfloor - 1} (|D| - s - i))^{m}$.

When $c$ is known. Given $s$ and $m$, the expected number of oracle calls made by the sample and probe strategy is $EOC(s, m) = E[|S|] = |D|(1 - \frac{1}{|D|^m})$. When $c$ is known, we can determine $s = s^*$ and $m = m^*$, which minimizes $EOC(s, m)$ while ensuring $f(|D|, s, m) \geq 1 - \delta$, by solving the following equation:

$$\min_{s, m} EOC(s, m) = |D||(1 - \frac{s}{|D|^m})$$

$$s.t. \quad f(|D|, s, m) \geq 1 - \delta \quad (8)$$

By plugging in the expression for $f(|D|, s, m, c)$, the constraint can be simplified to $m \geq \lfloor \log(1 - \frac{1}{|D|^m}) \rfloor$. By denoting the RHS as $m(s)$, we can rewrite the constraint as $m \geq m(s)$ for simplicity. Note, for a given $s$, $EOC(s, m)$ monotonically increases as $m$ increases. For a fixed $s$, the optimal $m$ which ensures a high success probability (i.e., $\geq 1 - \delta$) and minimizes $EOC(s, m)$ is clearly, $m = m(s)$. As a special case, we have $m^* = m(s^*)$. Thus, a naive approach for finding $s^*$ and $m^*$ is to compute $EOC(s, m)$ for each $1 \leq s \leq |D|$ and $m = m(s)$ and picking the best.

Such exhaustive search for the exact value of $s^*$ and $m^*$, however, can be expensive in a large DB. Instead, we are interested in approximation solutions with good guarantees, which we develop next. Given a query, let $(s, m)$ denote the sample size and number of samples used by a strategy. Then $|D| - EOC(s, m)$ denotes the expected number of saved oracle calls compared with the exhaustive approach of probing every object in the DB. Define the savings ratio as $\xi(s, m) = \frac{|D| - EOC(s, m)}{|D| - EOC(s^*, m^*)}$. It denotes the fraction of oracle calls saved by strategy $(s, m)$ compared to the optimal strategy $(s^*, m^*)$.

A larger $\xi$ indicates a better approximation, and the optimal strategy $(s^*, m^*)$ yields $\xi(s^*, m^*) = 1$.

Let us examine the special cases where either $s = 1$ or $m = 1$. For $s = 1$, we let $m = m(1)$, and

$$\xi_{s=1} := \xi(1, m(1)) \geq \frac{\delta}{\delta + 1} \cdot (1 - 1/|D|) \quad (9)$$

For $m = 1$, we set $s = s_1 := \lfloor \frac{-\log(\delta)}{log(\delta)} \rfloor$ to ensure high success probability, and

$$\xi_{m=1} := \xi(s_1, 1) \geq \frac{\delta}{\delta + 1} \cdot (1 - 1/|D| + \log(\delta)/\epsilon) \quad (10)$$

In practice where e.g., $\delta = 0.1$, $|D| = 10,000$, and $\epsilon = 100$, we have both $\xi_{s=1}$ and $\xi_{m=1}$ being no less than 97.7%, that is, if we fix either $s = 1$ or $m = 1$ as above, the saved oracle usage is at least 97.7% of what the optimal strategy $(s^*, m^*)$ achieves. Thus, either of them can be used as an approximation to the optimal strategy.

When $c$ is unknown. We incur extra oracle calls and apply Hoeffding Bounds [50] to ensure high success probability.

Proposition 4.2 (Hoeffding Bounds). Let $X_1, \ldots, X_n$ be independent random variables, with $X_i \in [0, 1] and let $E[X_1] = \mu$. Let $\mu = \frac{1}{n} \sum_{i=1}^n X_i$. Then for $\forall \epsilon \geq 0$, we have the concentration bound

$$Pr[|\mu - \epsilon| \leq \mu] \leq 1 - \exp(-2n\epsilon^2) \quad (11)$$

For an RT query with target $y$, we can derive a probabilistic lower bound for $s$ as follows. For an RT query, the core set $C$ consists of the top $(1 - \gamma) \times 100\%$ oracle neighbors of largest proxy indices. That is, we can write $c = \lfloor |NC| (1 - \gamma) \rfloor + 1$. When $s$ and $m$ are fixed, $f(|D|, s, m, c)$ monotonically increases as $c$ increases. Given $\delta_1 \in (0, 1)$, let $c$ denote a probabilistic lower bound of $c$, i.e., $Pr[c \geq \hat{c}] \geq 1 - \delta_1$. We can solve Eq. 8, either exactly or approximately as needed, subject to a more stringent constraint $f(|D|, s, m, c) \geq 1 - \delta_1$ to find $s$ and $m$, which ensures an overall success probability no less than $1 - \delta_1$. We show how to derive such probabilistic lower bound $\hat{c}$ using Hoeffding Bounds.

Randomly draw $x_i \in D$. Define $X_i = 1$ if $\text{dist}^O(x_i) \leq \epsilon$. We have $\mu_\delta := E[X_i] = \frac{|NC\langle y \rangle|}{|D|}$. Randomly draw $X_i \in [0, 1]$ with replacement. Denote $\mu_D = \frac{1}{n} \sum_{i=1}^n X_i$. For any $\epsilon, \delta_1 \in (0, 1)$, by Hoeffding Bounds, we have $Pr[|\mu_\delta - \mu| \leq \mu_D] \geq 1 - \delta_1$ if $n \geq \log(\delta/\epsilon)$. For an RT query with target $y$, since $c = |NC\langle y \rangle (1 - \gamma) | + 1$, and we have $Pr[c \geq |D|(\mu_D - \epsilon)(1 - \gamma)] + 1 \geq 1 - \delta_1$, if $n \geq \log(\delta/\epsilon)$. We denote the probabilistic lower bound $\hat{c} := |D|(\mu_D - \epsilon)(1 - \gamma) + 1$.

For PT queries, such $\hat{c}$ is hard to obtain. Given $\delta$ and $k$, we apply Hoeffding Bounds in a similar way to derive a probabilistic lower bound for the precision $M_p(D_k)$, denoted as $\mu_M$. That is, $Pr[M_p(D_k) \geq \mu_M] \geq 1 - \delta_1$. For a PT query with target $y$, $D_k$ is a high probability valid answer if $\mu_M \geq \gamma$. We use heuristics to identify $D_k$ of high $\mu_M$ and good $\mathcal{C}$ (details in §4.2.2).
4.2.2 Algorithms with Core Set Closure.

Algorithm CSC. Algorithm CSC returns high probability valid answers with a minimal expected number of oracle calls and empirically good CR, under Core Set Closure and taking c as input. CSC is presented in Algorithm 3. We compute \( s^* \) and \( m^* \) in line 2 either exactly or approximately, and draw samples in line 3. In lines 4 to 8, we compute \( k_S \) according to the query type and return \( D_{k_S} \) as the answer. The exact solution to Eq.8 requires \( O(c|D|) \) operations while approximate solutions take \( O(D) \) operations. The time complexity is dominated by accessing proxy prefixes, which requires sorting all objects w.r.t. proxy distance taking \( O(D|\log(D)|) \).

Algorithm CSC

Algorithm 3: CSC

1. Function CSC\((D, c, \delta)\):
2. \( s^*, m^* \leftarrow \text{getsm}(|D|, c, \delta) \) /* Solve Eq.8 */
3. \( S \leftarrow \text{UniformSample}(D, s^*, m^*) \)
4. if RT query then
5. \( k_S \leftarrow \max\{I(x) \mid x \in S \land \text{dist}^D(x) \leq r\} \)
6. else
7. \( k_S \leftarrow \min\{I(x) \mid x \in S \land \text{dist}^O(x) \leq r\} \)
8. return \( D_{k_S} \)

Algorithm CSE. Algorithm CSE incurs more oracle calls and returns high probability valid answers in general settings where \( c \) is unknown or Core Set Closure assumption does not hold.

CSE is presented in Algorithm 4. CSE-RT is given in lines 1 to 4. Given \( r, \delta \), we sample and probe \( n = \left\lfloor \frac{\log(\delta)}{2r} \right\rfloor \) objects to derive \( \tilde{c} \). Then, we invoke Algorithm CSC to process the query subject to \( f(|D|, s, m, c) \geq \frac{1-\delta}{1-\delta} \). CSE-PT is given in lines 5 to 15. We use CSC to find good answer candidates, and apply Hoeffding Bounds to return high probability valid answers. Specifically, given a query and budget \( b' \), we sample and probe \( b' \) objects to estimate \( c \), then invoke CSC with the estimation to compute \( D_{k'} \) (lines 6 to 8). We also use the same sample to estimate the largest \( k = k_S \) such that \( D_k \) has a sampled precision no less than \( y \) (line 9). To improve CR (i.e., recall for PT queries), we set \( \tilde{k} = \max\{k_1, k_2\} \) and consider \( D_{\tilde{k}} \) as the answer candidate (line 10). In lines 11 to 15, given \( e_p \), we draw samples and estimate a probabilistic lower bound for \( M_p(D_{\tilde{k}}) \) by applying Hoeffding Bounds. For a PT query with target \( y \), we return \( D_{\tilde{k}} \) if the probabilistic lower bound is no less than \( y \). O/w, we return all oracle neighbors identified from samples. The overall time complexity is dominated by CSC and is also \( O(D|\log(|D|)) \).

Algorithm 4: CSE

1. Function CSE-RT\((D, \delta)\):
2. \( \hat{\mu}_D \leftarrow \text{HoeffdingEst}(D, \delta, r) \)
3. \( \zeta \leftarrow |D|\hat{\mu}_D - (1-\gamma) + 1 \)
4. return CSC\((D, \zeta) \)

5. Function CSE-PT\((D, \delta)\):
6. \( S \leftarrow \text{UniformSample}(D, b') \)
7. \( \hat{c} \leftarrow \frac{|D|}{|S|} \cdot \text{the size of core set w.r.t. } S \)
8. \( D_{k_1} \leftarrow \text{CSC}(D, \hat{c}, 1 - \delta) \)
9. \( k_2 \leftarrow \max\{I(x) \mid x \in S \land M_p(D_{k_1} \cap S) \geq \gamma\} \)
10. \( \hat{k} \leftarrow \max\{k_1, k_2\} \)
11. \( \mu_{D_{\hat{k}}} \leftarrow \text{HoeffdingEst}(D_{\hat{k}}, \delta, e_p) - e_p \)
12. if \( \mu_{D_{\hat{k}}} \geq \gamma \) then
13. \( \text{return } D_{\hat{k}} \)
14. else
15. \( \text{return } \{x \in S \mid \text{dist}^O(x) \leq r\} \)

6. Function HoeffdingEst\((D, \delta, e)\):
17. \( S \leftarrow \text{UniformSample}(D, \left\lfloor \frac{\log(\delta)}{2e}\right\rfloor) \)
18. return \( \hat{\mu} = \frac{\sum_{x \in S} \text{dist}^H(x) \leq r}{|S|} \)

5 EXPERIMENTS

Our extensive experiments (1) assess the performance of PQA to demonstrate its optimality w.r.t. CR and success probability under Proxy Quality assumption (§ 5.2), (2) assess the performance of CSC to demonstrate its minimal oracle usage and success probability under Core Set Closure assumption (§ 5.3), (3) compare PQE, CSE with the baselines on CR and success probability under the same oracle usage (§ 5.4) and under varied oracle budgets (§ 5.5), (4) Compare PQE, CSE with the baselines w.r.t. query time (§ 5.6), (5) Compare PQE, CSE with the baselines w.r.t. CPU overhead, CR, and success probability on datasets of various sizes and domains (§ 5.7).

5.1 Experimental Setup

5.1.1 Datasets and Proxy Models.

Multi-label Image Recognition. VOC [20] and COCO [36] are widely used benchmarks in multi-label recognition tasks. The validation set of COCO consists of 40,504 images from 80 classes, and VOC contains 4,952 images from 20 object categories. We also uniformly sample a 8000-image subset from COCO, denoted as COCO (small). We use VOC and COCO (small) in different experiments.

Medical. Mimic-III [27] and elCu [42] are two publicly available clinical datasets, that include patient trajectories, demographics collected by daily ICU admissions, and clinical measurements. After pruning records with only one admission, we obtain a Mimic-III subset of 4,243 records and an elCu subset of 8,235 records.
Video. We use the night-street dataset [10] to support queries over classification tasks. Each video frame has a Boolean label indicating whether or not it contains a car. We uniformly draw a subset of 10,000 frames from the original dataset for evaluation.

5.1.2 Baselines. We consider the following baselines.

SUPG The closest work to ours is SUPG [30]. SUPG uses oracle $O'$ with a Boolean output and a proxy model $P'$ which outputs a score in $[0, 1]$. Given a query object $q$ and a radius $r$, our problem can be mapped to a binary classification problem: for each object $x$, is it a near neighbor to $q$ w.r.t. $r$? Given oracle $O$ and proxy $P$ for our problem, a natural oracle predicate for SUPG should output 1 when the given object is a near neighbor and 0 otherwise. This translates to $O'(x) = 1$ iff $dist^O(x) < r$. Similarly, a natural proxy model for SUPG should give high scores when the corresponding object is more probable to be a near neighbor. As illustrated in Figure 5, with a properly chosen proxy, proxy distance $dist^P(x)$ is a good approximation for oracle distance $dist^O(x)$. Given that $dist^P(x) \in [0, 1]$ in our problem, we choose $P'(x) = 1 - dist^P(x)$ as the proxy model for SUPG. Intuitively, if object $x$ has a small proxy distance, $x$ is more likely to have a small oracle distance as well and hence more probable to be classified as $O'(x) = 1$, which is properly reflected by a high value of $P'(x)$.

Probabilistic Top-K [33]. This baseline studies approximate Top-K queries and delivers solutions with statistical guarantees. Given a query, there exists a direct mapping from our FRNN query to a Top-K query. For example, given a query object $q$ and radius $r$, an FRNN query asks for an answer $Ans$ which comprises all near-neighbours within the radius $r$ to $q$. Naturally, we can rewrite this query in Top-K semantics: given query object $q$, return the Top-$K$ nearest neighbors to $q$ where $K = |Ans|$ according to the aforementioned FRNN query. Furthermore, this Top-K baseline relies on distribution over oracle predictions, which can be obtained from our Proxy Quality assumption in PQA.

Sample2Test Given an FRNN RT (resp. PT) query, this baseline first probes samples w.r.t. a given oracle budget, and then selects the optimal prefix as the answer according to sample precision (resp. recall). This is the approach used in probabilistic predicates (PP) [38], NoScope [29], and also serves as a baseline in SUPG [30]. Given a sample $S \subset D$ and a proxy index $k$, denote $S^k = S \cap D_k$. The sample precision at $k$ is $Precisions(k) = \frac{|S^k \cap NN_O^k|}{|S^k|}$ and the sample recall is $Recalls(k) = \frac{|S^k \cap NN_O^k|}{|NN_O^k|}$. Given dataset $D$ and the target $y$, this baseline returns $D_k$ where $k' = \max(1 \leq k \leq |D| \mid Precisions(k) \geq \gamma)$ for PT queries, and $k' = \min(1 \leq k \leq |D| \mid Recalls(k) \geq \gamma)$ for RT queries. We select the largest (resp. smallest) proxy prefix for PT (resp. RT) to improve CR.

Scan2Test We also consider the naive approach which probes all objects with the oracle and selects the correct answer set for a given query. This approach is used as the baseline in [33].

5.1.3 Evaluation Measures. For both RT and PT queries, we are interested in three measures: (i) empirical success probability in relation to the required success probabilities; (ii) average CR of answers returned by different methods; and (iii) query processing time including (a) CPU overhead and (b) number of oracle calls. We do not compare proxy time since it is identical for all approaches and is only a fraction of the overall query processing time.

5.1.4 Protocol. Our evaluation protocol randomly chooses several query objects from a dataset and aggregates our measures for those query objects. In Section 5.2, we randomly choose 200 query objects and aggregate their results. In other experiments, we randomly choose 50 query objects and execute each query 10 times and aggregate their results. This is because PQA is deterministic while other algorithms are subject to randomness, so we average over multiple trials. We use cosine distance whenever the model outputs are multi-dimensional vectors: $dist_{cos}(y_1, y_2) = 1 - \cos(y_1, y_2)$ given its wide application in proximity query processing [2, 32, 39]. When the output is scalar (e.g., Boolean labels), we use the absolute difference $dist_{abs}(y_1, y_2) = |y_1 - y_2|$ as the distance function, which allows us to generalize SUPG query with boolean oracle predicates.

In all cases, the radius threshold is $r = 0.9$. The choice of distances and thresholds has no impact on our statistical guarantees.

Default values. Unless otherwise stated, we set $\gamma$ (recall and precision targets) to 0.95 and $\delta$ to 0.1 in all our experiments. We add a black dashed line (---) at the level of 1 $- \delta$ in figures to help visually track the success probability of each approach. We empirically choose $\sigma_0 = 0.3$ for PQE, $\epsilon_p = 0.1\%$ and $b' = 100$ for CSE-PT, $\epsilon_r = 10\%$ and $\delta_r = 0.05$ for CSE-RT according to our experiment results. We choose a small $\epsilon_p$ for CSE-PT to improve the probabilistic lower bound for precision, and a relatively large $\epsilon_r$ for CSE-RT to reduce the oracle usage incurred by applying Hoeffding Bounds. In addition, we only report results of CSE when $m = 1$ (see Eq. 10) given its dominating performance over other m settings.

Our algorithms are implemented in Python 3.7 and experiments are conducted on a M1 Pro chip @ 3.22GHz with a 16GB RAM.

5.2 PQA Success at Maximal CR

PQA finds high probability valid answers of maximal expected CR with zero oracle calls, whenever Proxy Quality assumption holds ($\S$4.1). We test it on two semi-synthetic datasets. Specifically, we use real proxy distances from VOC and eICU, and synthesize oracle distances with a normal distribution $N(0, \sigma = 0.1)$. We clip the normal distribution to $[0, 1]$ to agree with the output range of our distance measures. We demonstrate CR maximality and success probability guarantees of PQA by comparing it with a series of variants. Recall that PQA returns the top- $k^*$ objects of smallest proxy distances as the answer. We measure the success probability and CR when using a perturbed $k^*$. We try perturbations ranging from $-20\%$ to $20\%$ by returning top-$(1 + perturb) \cdot k^*$ for $-20\% \leq perturb \leq 20\%$.

The results are shown in Figure 7. The two top plots summarize RT queries. Recall $\delta = 0.1$, which requires success probability being no less than $90\%$. On VOC, with zero perturbation, PQA achieves a $92\%$ empirical success probability and $39\%$ CR for RT queries. On eICU, the empirical success probability and CR are $90\%$ and $38\%$.

| Table 3: Datasets Summary |
|----------------------------|
| **Datasets** | **Oracle** | **Proxy** | **Query targets** |
| VOC&COCO | Human labeler | ML-GCN[14] | Similar images |
| Mimic-Ill&eICU | Physicians | Lig-Doctor[46] | Similar patients |
| night-street | Mask R-CNN[24] | ResNet-50[25] | Car frames |
respectively. With negative perturbation, empirical success probability quickly shrinks to nearly zero; with positive perturbation, CR starts to drop. This observation clearly demonstrates that PQA gives the highest answer CR while respecting the success probability constraint. The two bottom plots are for PT queries, and are similar to RT. The unperturbed PQA achieves 94% empirical success probability on both datasets, a 53% CR on VOC, and a 42% CR on eICU. Any perturbation to \( k^* \) either fails the success probability constraint or degrades CR.

We compare PQA and Top-K on the same semi-synthetic VOC dataset (see Figure 8). Both methods achieve desired success probability targets. However, Top-K suffers from huge oracle usage while PQA needs no oracle calls, indicating PQA is capable of efficient query processing when proxy quality distribution is known. Furthermore, we investigate the sensitivity of PQA to \( \sigma^2 \), shown in Figure 9. We test PQA on VOC with various \( \sigma \) values and report PQE performance\(^3\) for comparison purposes. As \( \sigma \) increases, for both query types, the success probability of PQA increases while CR decreases. This agrees with the intuition that, as the proxy quality gets worse, PQA becomes more conservative to improve success probability at the cost of CR degradation. Since PQE does not rely on external \( \sigma \), both success probability and CR are constant and higher than PQA, given that PQE has the flexibility to probe samples with the oracle.

5.3 CSC with Minimal Oracle Usage

CSC ensures high success probabilities with minimal oracle usage under Core Set Closure assumption (§ 4.2). We implement an exact algorithm and two approximation algorithms to compute \( s^* \) and \( m^* \) (§ 4.2.1). \( \text{Approx-s1} \) and \( \text{Approx-m1} \). We compare these algorithms to two baselines, \( \text{Rand-s} \) and \( \text{Rand-sm} \). Specifically, \( \text{Rand-s} \) randomly chooses \( s \) and sets \( m = m(s) \), whereas \( \text{Rand-sm} \) chooses both \( s, m \) at random. For each query, we precompute the core set size and feed it to all approaches. We study the empirical success probability and oracle usage on VOC and eICU. The results for RT and PT queries are reported in Figure 10. Especially, we report standard deviation of oracle usage for both query types on both datasets. We also report CPU overheads in Figure 11 (Left).

All approaches achieve high empirical success probability for both query types. For RT queries, the exact algorithm invokes the oracle on only 9.8% objects in VOC and 7.1% objects in eICU. The oracle usage of approximation algorithms is just up to 1.1% more than the exact algorithm. However, the baseline \( \text{Rand-s} \) applies the oracle on at least 49.3% objects and \( \text{Rand-sm} \) makes oracle calls on at least 94.6% objects. For PT queries, the exact algorithm has the smallest oracle usage, which accounts for 8.1% objects in VOC and 5.2% objects in eICU. The approximation algorithms incur an oracle usage which is just up to 2.1% higher than the exact algorithm. The baseline method \( \text{Rand-s} \) calls the oracle on at least 39% objects, and \( \text{Rand-sm} \) makes oracle calls for at least 93.7% objects.

We are also interested in the CPU overhead of the exact algorithm and the two approximation algorithms. Results on our five real-world datasets are summarized in Figure 11 (Left). Clearly, the exact algorithm has a larger CPU overhead in comparison to the two approximation algorithms. Specifically, the approximation algorithms achieve a speedup up to 1466× for RT queries and 7391× for PT queries on CPU overheads, in comparison to the exact method.

On VOC, we investigate how sensitive CSC is to the input core set size \( c \) and include CSE performance for the same query for comparison. Results are summarized in Figure 11 (Right). As \( c \) increases, for both query types, success probability of CSC decreases and CR increases. Since CSE estimates \( c \) internally, both success probability and CR are agnostic to external \( c \) changes. Note that the CSE performance is generally better than CSC, which is attributed to the fact that CSE has more flexibility to probe objects with oracle for the additional \( c \) estimation.

5.4 CSE and PQE vs SUPG and Sample2Test

We implement CSE and PQE and compare them to SUPG and Sample2Test. For a fair comparison, we use the oracle usage incurred by CSE as the budget for PQE, SUPG, and Sample2Test. We measure empirical success probability and CR on VOC and eICU (Figure 12). For RT, CSE and PQE achieve high empirical success probability on both datasets, while SUPG fails the success threshold empirically by a margin of 10% on VOC. On CR, PQE outperforms SUPG by a margin up to 26%, and CSE outperforms SUPG up to 33%. For PT,
all approaches achieve high empirical success probability. On CR, PQE outperforms SUPG by up to 12% and CSE achieves comparable CR to SUPG. Sample2Test continuously fails the success probability for both query types on both datasets.

Failures of SUPG on RT queries stem from the sample mean and variance it uses without error bounds, which introduces uncontrolled uncertainty and degrades statistical guarantees.

5.5 Oracle Efficiency
To measure Oracle efficiency, we perturb the oracle usage incurred by CSE and use it as the budget for PQE, SUPG, Sample2Test. The CR of CSE is also plotted as a baseline. Results are in Figure 13. For RT queries, PQE achieves high empirical success probability on both datasets, while SUPG and Sample2Test fail frequently on VOC especially with small budgets. This indicates that CSE is the most oracle efficient approach for RT queries. For PT queries, all approaches except Sample2Test achieve high empirical success probability on both datasets.

5.6 Time Efficiency
The running time of a query is composed of CPU overhead and model usage including proxy and oracle calls. We measure CPU overhead for each approach locally and approximate model usage by timing the number of model calls and average time taken by each call. For instance, on medical datasets (MIMIC-III & eICU), the oracle
is a human physician whose average diagnosis time is 15 minutes [47], while the proxy is a RNN model taking 1 ms for each call [46]. Results are reported in Table 4, 5 with the best results in bold and saving ratios w.r.t. SUPG. For both query types, Sample2Test and Scan2Test are the two most time-consuming approaches.

### 5.7 Scalability

We measure CPU overhead, success probability, and CR of PQE, CSE, SUPG, and Sample2Test. We uniformly draw subsets of the original COCO dataset (25%, 50%, 75%, and 100%). To make a fair comparison, we use the oracle usage incurred by CSE as the budget for PQE and SUPG. Results are shown in Figure 14 and Table 6, 7. For both query types, CSE, SUPG, and Sample2Test have a reasonably low CPU overhead, while PQE has the highest CPU overhead, 2.5 and 43.7 seconds per query for PT and RT queries separately.

### 6 RELATED WORK

#### Query approximation

Query approximation techniques [34] can be categorized into (1) online aggregation: select samples online and use them to answer OLAP queries, and (2) offline synopses generation to facilitate OLAP queries. Our work adopts a probabilistic top-k approach [48] and is significantly different from these.

**FRNN query.** FRNN query answering systems [5, 6] build spatial indexes on the whole DB, which requires oracle calls on every single object. Our work focuses on reducing the oracle usage and is clearly distinguished from this line of work.

**Optimizing ML inference.** Several recent approaches were proposed to speed up the application of an ML model. Existing approaches follow either an in-database [16] or in-application approach [1]. Amazon Aurora is an example of an in-database containerized solution that enables external calls from SQL queries to ML models in SageMaker [17]. Containerized execution introduces overhead in prediction latency. To mitigate that, Google’s BigQuery ML [21] and Microsoft’s Raven were developed [31]. Compared to Raven, BigQuery ML relies mostly on hard-coded models and targets batch predictions, since it inherits a relatively high startup cost. Raven and its runtime environment ONNX [12] offer the additional ability to make tuple-level inference.

### 7 CONCLUSION AND DISCUSSION

We formalize and solve precision-target and recall-target queries, two paradigms that are well-suited for querying the results of ML predictions. We propose two assumptions and develop four algorithms. Our extensive experiments on five real-world datasets show that our approach enjoys statistical guarantees with a small cost and a good complementary rate, i.e., a good balance between recall and precision rates. Our framework can be extended to optimize a query workload using metric properties like triangle inequality [4]. Consider the objects \( \{ q_1, q_2 \} \). Suppose that we first choose \( q_1 \) as the query object, and compute the proxy distances \( \text{dist}^P(q_1, q_2) \) and \( \text{dist}^P(q_1, x) \) to find answers using our approaches. Next, when we choose \( q_2 \) as the query object, by leveraging triangle inequality, we can lower bound \( \text{dist}^P(q_2, x) \) as \( \text{dist}^P(q_2, q_1) \geq |\text{dist}^P(q_1, q_2) - \text{dist}^P(q_1, x)| \). If this bound is high, we can safely avoid applying the probe to \( x \) for query \( q_2 \). We can extend our framework to multiple proxies at different accuracy and cost levels. This represents real-world scenarios where proxies are derived from huge neural models by activating specific subnetworks [8]. It is clear that, with multiple proxy models, the search space for our optimization problem will exponentially increase. We are currently exploring possible solutions.
REFERENCES

[1] Zeeshan Ahmed, Saed Amidehaz, Mikhail Bilenko, Rogan Carr, Wei-Sheng Chin, Yael Dekel, Xavier Dupré, Vadim Eksarevsky, Senji Filipi, Tom Finley, Ahsheek Goorwani, Monte Hoover, Scott Ingalls, Matteo Interlandi, Naejeh Kazmi, Gerk Krivosheev, Pete Luferenko, Ivan Matansey, Sergey Matusevych, Shahab Moradi, Gani Nazirov, Justin Ormont, Gal Oshri, Artidoro Pagnoni, Jignesh Parmar, Prabhat Roy, Mohammad Zeeshan Siddiqui, Markus Wesmer, Shauheen Zarrinazam, and Ying Zhu. 2019. Machine Learning at Microsoft with ML.NET. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019, Ankur Teredesai. Vipin Kumar, Ying Li, Rómer Roesales, Evimaria Terzi, and George Karypis (Eds.). ACM, 2448–2458.

[2] Mohammad Alodadi and Vandan P. Janeja. 2015. Similarity in Patient Support Systems Design and Implementation, OSDI 2018, Carlsbad, CA, USA, October 8-10, 2018. Optimizing Compiler for Deep Learning. In Proceedings of the 2018 International Conference on Management of Data, SIGMOD ’18, Association for Computing Machinery, New York, NY, USA, Article 2, 7 pages. https://doi.org/10.1145/3393957.3399870

[18] Stanislaw Denizziak and Tomasz Michno. 2016. Content based image retrieval using query by approximate shape. In 2016 Federated Conference on Computer Science and Information Systems (FedCSIS). 807–816.

[19] Dujian Ding, Shemer Amer-Yahia, and Laks VS Lakshmanan. 2022. On Efficient Approximate Queries over Machine Learning Models. https://doi.org/10.48550/ARXIV.2206.09415

[20] Mark Everingham, Luc Van Gool, Christopher I. K. Williams, John Winn, and Andrew Zisserman. 2010. The Pascal Visual Object Classes (VOC) Challenge. International Journal of Computer Vision 88, 2 (2010), 303–338.

[21] Sriraj Fernandes and Jorge Bernardino. 2015. What is BigQuery?. In Proceedings of the 19th International Database Engineering & Applications Symposium (Yokohama, Japan) (IDEAS ’15). Association for Computing Machinery, New York, NY, USA, 202–203. https://doi.org/10.1145/2790557.2790977

[22] Junyang Gao, Yifan Xu, Pankaj K. Agarwal, and Jian Yang. 2021. Efficiently Answering Durability Prediction Queries. In SIGMOD ’21: International Conference on Management of Data, Virtual Event, China, June 20–25, 2021. 591–604.

[23] Geoffrey R. Grimmett. 1968. Probability: An Introduction. Oxford University Press.

[24] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2017. Mask R-CNN. CoRR abs/1706.03768 (2017). arXiv:1706.03768 http://arxiv.org/abs/1706.03768

[25] Kaiming He, Kevin Gokhale, Piotr Dollár, and Ross Girshick. 2017. Mask R-CNN. In Proceedings of the 2018 Conference on Computer Vision and Pattern Recognition (CVPR) CoRR abs/1803.08890 (2018). arXiv:1803.08890 [cs.CV]

[26] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2017. Mask R-CNN. In Proceedings of the 2018 Conference on Computer Vision and Pattern Recognition (CVPR) CoRR abs/1803.08890 (2018). arXiv:1803.08890 [cs.CV]
