Semantic Indexes for Machine Learning-based Queries over Unstructured Data[*]

Daniel Kang*
Stanford University
Stanford, USA
ddkang@stanford.edu

John Guibas*
Stanford University
Stanford, USA
jtguibas@stanford.edu

Peter Bailis
Stanford University
Stanford, USA
pbailis@stanford.edu

Tatsunori Hashimoto
Stanford University
Stanford, USA
thashim@stanford.edu

Matei Zaharia
Stanford University
Stanford, USA
matei@cs.stanford.edu

ABSTRACT

Unstructured data (e.g., video or text) is now commonly queried by using computationally expensive deep neural networks or human labelers to produce structured information, e.g., object types and positions in video. To accelerate queries, many recent systems (e.g., BlazeIt, NoScope, Tahoma, SUPG, etc.) train a *query-specific proxy model* to approximate a large *target labelers* (i.e., these expensive neural networks or human labelers). These models return *proxy scores* that are then used in query processing algorithms. Unfortunately, proxy models usually have to be trained *per query* and require large amounts of annotations from the target labelers.

In this work, we develop an index (trainable semantic index, TASTI) that simultaneously removes the need for per-query proxies and is more efficient to construct than prior indexes. TASTI accomplishes this by leveraging semantic similarity across records in a given dataset. Specifically, it produces embeddings for each record such that records with close embeddings have similar target labeler outputs. TASTI then generates high-quality proxy scores via embeddings without needing to train a per-query proxy. These scores can be used in existing proxy-based query processing algorithms (e.g., for aggregation, selection, etc.). We theoretically analyze TASTI and show that a low embedding training error guarantees downstream query accuracy for a natural class of queries. We evaluate TASTI on five video, text, and speech datasets, and three query types. We show that TASTI's indexes can be 10× less expensive to construct than generating annotations for current proxy-based methods, and accelerate queries by up to 24×.

ACM Reference Format:
Daniel Kang*, John Guibas*, Peter Bailis, Tatsunori Hashimoto, and Matei Zaharia. 2022. Semantic Indexes for Machine Learning-based Queries over Unstructured Data. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

[*]Marked authors contributed equally.

1 INTRODUCTION

Unstructured data, such as video and text, is becoming increasingly feasible to analyze due to deep neural networks (DNNs). A common approach is to use DNNs to extract structured information from these sources and use the structured data to answer queries. However, accurate DNNs can be prohibitively expensive to execute on large data volumes. For example, Mask R-CNN [23] (an object detection DNN) can be used to extract object types and positions from frames of video, which can be used to answer queries such as counting the number of cars visible in a frame or finding frames with both a car and a bicycle. Unfortunately, Mask R-CNN executes as slow as 3 frames per second (fps), or 10× slower than real time.

To reduce these costs, recent work has proposed *query-specific proxy models* to approximate high-quality *target labelers*. Low-cost proxy models can be used for selecting data records that match a predicate, aggregation queries, and limit queries [3, 7, 26, 31–33, 38]. For each query, a proxy model is trained to generate *proxy scores* for data records, in which the goal is to approximate the result of executing the target labeler on that data record for the particular query. These scores are then used in various algorithms depending on query type. For example, BlazeIt [31] will train a proxy model to (approximately) count the number of cars per frame of a video to answer the car counting query, and selection algorithms (e.g., NoScope [32], probabilistic predicates [38], SUPG [33], and Tahoma [3]) will train a separate proxy model to (approximately) filter frames with cars and bicycles for selecting such frames.

Unfortunately, methods based on query-specific proxy models have three key drawbacks. First, obtaining large amounts of training data from the target labeler to train the proxy models can be expensive. For example, BlazeIt and NoScope require hours of GPU compute to execute the target labeler to produce labels to train the proxy models [31, 32] and other systems require expensive human annotations [3, 26, 38]. Second, these systems require new training procedures for each query type, which can be difficult to develop. Third, query-specific proxy models cannot easily share computation across different queries or query types. Thus, using proxy models can be challenging and computationally expensive.

We observe that this prior work ignores a key opportunity: redundancy present in the *target labeler outputs* of many datasets. For example, two frames with visually distinct cars in the bottom left would have the same result for many queries, e.g., counting the number of cars or selecting cars in the bottom left. Namely, the structured outputs (i.e., target labeler results) of many data records are semantically similar. While fully computing target labeler outputs for all records is expensive, query processing systems would ideally use this similarity to avoid repeated work and target labeler invocations.

To address these issues and leverage this opportunity, we propose **T**r**A**inable **S**eman**T**ic **I**ndexes (TASTI). TASTI is an indexing method over unstructured data for accelerating downstream proxy score-based query processing methods via embeddings (i.e., vectors in $\mathbb{R}^n$). Given the target labeler and a user-provided closeness function over target labeler outputs, TASTI produces embeddings for each unstructured data record (e.g., frame of video), with the desideratum that close records have close embeddings. TASTI's required closeness function is often easy to specify, e.g., that frames of a video with similar object types and object positions are close (Section 3).[1] TASTI then uses the embeddings and a small set of records annotated by the target labeler to answer downstream queries.

Specifically, we propose a method of using TASTI's embeddings and the labeled records (i.e., cluster representatives) to generate proxy scores *automatically*, including for proxy-based aggregation, selection, and limit query processing algorithms (Section 4) [3, 31–33, 38]. TASTI generates per-record proxy scores by propagating annotations from the cluster representatives to the unlabeled records. For example, we could assign an unannotated frame the number of cars in the closest cluster representative for counting cars. These scores can then used in query processing algorithms, such as those in BlazeIt, probabilistic predicates, Tahoma, etc. Moreover, as the target labeler is executed over more data during query processing, we can incrementally improve TASTI's clustering, which improves performance (i.e., TASTI's indexes support "cracking" [27]).

To understand TASTI's performance, we provide a theoretical analysis of TASTI and downstream query accuracy (Section 5). We prove that queries that are Lipschitz-continuous functions of the data will achieve *exact* results when using TASTI (with sufficiently dense clustering) under 0 training loss, and quantitative bounds when the training loss is not 0. Although our assumptions are strong, our analysis provides statistically grounded intuition for why TASTI can outperform baselines. We validate our intuition with extensive experiments (Section 6).

We implemented TASTI in a prototype system and evaluated it on four datasets, including widely studied video datasets [11, 29, 31, 33, 43], a text dataset [46], and a speech dataset [4]. We integrated TASTI into query processing algorithms for aggregation, selection, and limit queries as proposed by prior work and executed these queries over the datasets. We show that TASTI's indexes require up to 10× *fewer* labels from the target labeler to construct than generating training data for per-query proxy model methods, as TASTI leverages redundancy in the datasets. Furthermore, TASTI outperforms on query runtime across all queries and datasets we evaluate on by up to 24× over previous optimized systems.

In summary, our contributions are:

(1) We propose a method for constructing semantic indexes (TASTI) for queries over unstructured data.
(2) We theoretically analyze TASTI, providing a statistical understanding of our algorithm's performance.
(3) We evaluate TASTI on five datasets and three query types, showing it can outperform state-of-the-art.

[1]TASTI can also be used without training an embedding by using pre-trained embeddings, although query performance will generally be better with its training method.

2 OVERVIEW AND EXAMPLE

2.1 Background and Problem

We first describe how target labelers and proxy scores are used in analytics systems before describing an overview of TASTI.

Many analytics applications over unstructured data are powered by expensive DNNs that extract structured data from these unstructured sources or human labelers (e.g., ground-truth annotations for studying social or life sciences [34]). We refer to these expensive DNNs and human labelers as *target labelers*. These target labelers *induce a schema* over the extracted data. For example, object detection DNNs can extract information about object types and positions from frames of a video. The schema would contain columns for object type, object (x, y)-positions, and timestamps. Unfortunately these high-quality target labelers, such as Mask R-CNN or BERT, can be expensive and dominate query execution costs.

Thus, recent work attempts to accelerate queries with target labelers by using proxy scores (e.g., BlazeIt, NoScope, probabilistic predicates, Tahoma, SUPG, etc.). The most common method of producing proxy scores is to train a smaller DNN (also called a "specialized NN" or "proxy model") that will produce a proxy score per data record. These proxy models are typically trained to approximate the output of the target labeler for the query at hand, e.g., a count for an aggregation query, and can yield substantial query speedups. Constructing the training data can be expensive as the training data must reflect a wide range of potential queries.

We describe two examples of using proxy scores to accelerate queries, both of which train a new proxy model per query. In both cases (and more generally), the goal is to generate proxy scores that are highly correlated with the target labeler outputs: these algorithms will adaptively improve with better proxy scores.

Approximate aggregation. Suppose the user issues a query for the average number of cars per frame in a video, as studied by BlazeIt [31]. BlazeIt takes as input an error target and proxy scores.

To optimize this query, BlazeIt trains a cheap proxy model whose output is the predicted number of cars per frame using a sample of frames annotated by the target labeler. This proxy model is then used to generate a query-specific proxy score per frame. BlazeIt then uses these scores as a "control variate" [21] to reduce the variance in estimation. Proxy scores that are more correlated with the true count will result in faster query execution.

Approximate selection. Suppose the user issues a query to select 90% of frames with cars with 95% probability of success, as studied by the "recall target" setting in SUPG [33]. Specifically, SUPG takes as input a target recall and (in contrast to BlazeIt) a fixed target labeler budget. SUPG will train a proxy model that estimate the probability of a record matching a predicate.

Given proxy scores, SUPG will use importance sampling and return a set of records achieving the recall target. Other recent proxy-based systems accelerate selection queries without guarantees [3, 7, 26, 38].

2.2 TASTI's Inputs, Outputs, and Goals

Overview. As input, TASTI takes a target labeler, an induced schema, a target labeler invocation budget for index construction, a closeness function over the induced schema, and a parameter k that specifies how many distances to store for reach record. The primary cost in index construction are the target labeler invocations. TASTI will produce an embedding-based index subject to the budget that can produce proxy scores for a range of queries.

TASTI's primary goal is to produce high quality proxy scores for query processing algorithms, as with per-query proxy models, but *without* training a new model per query.

Supported queries. We demonstrate how to generate proxy scores for selection queries, aggregation queries, and limit queries [3, 31–33, 38] with TASTI. TASTI can be used with other queries requiring proxy scores. Since the initial draft, other work has used TASTI to support aggregation queries with predicates [34].

2.3 Example

Consider constructing an index for visual data, in which queries over object types and positions are issued. In this case, the target labeler (e.g., Mask R-CNN) takes an unstructured frame of video and returns a structured set of records that contains fields about the positions and types of objects in the frame. Consider two queries, one of which counts the number of cars per frame (aggregation query as supported by BlazeIt) and one that selects frames with cars (selection query as supported by NoScope, probabilistic predicates, SUPG, etc.). To understand how the index construction procedure and query processing works, we describe the intuition below.

Index construction. TASTI builds its index by training an embedding DNN for the input data and then clustering results based on it. Ideally, semantically similar records are grouped together, e.g., all frames with two cars might form one cluster, all frames with one bike and one car might form a cluster, etc.

To train an embedding DNN, TASTI requires a heuristic for "close" and "far" target labeler outputs, either as a Boolean function or as a cutoff based on a continuous distance measure. One such heuristic for our video application is to group frames with the same number of objects and similar positions together. The grouping of "close" frames can be specified in pseudocode as follows:

```
def is_close(frame1: List[Box], frame2: List[Box])->bool:
    if len(frame1) != len(frame2):
        return False
    return all_boxes_close(frame1 , frame2)
```

where `all_boxes_close` is a helper function that returns true if all boxes in `frame1` have a corresponding "close" box in `frame2`. Given the closeness function, TASTI trains a low-cost embedding DNN via the triplet loss [42], which separates "far" frames. Then, TASTI computes embeddings over all frames of the video with the embedding DNN and select a set of frames to annotate with the target labeler. It will then store the target labeler's outputs (object types and positions) for each cluster representative. TASTI uses the cluster representatives and distances from unannotated frames' embeddings for downstream query processing.

Query processing. TASTI can now be used to produce proxy scores for a range of downstream queries using existing proxy-based algorithms. First, TASTI generates exact scores on cluster representatives, e.g., the exact count of the number of cars from the cached target labeler outputs. Then, TASTI propagates these scores to the remainder of the records, e.g., producing approximate counts for the unannotated records. We describe the intuition behind two example queries: in both of these examples, TASTI need not train a new proxy model per query and can reuse its index.

Approximate aggregation. Consider the query of counting the average number of cars per frame. TASTI computes the query-specific proxy score as the distance-weighted average of the number of cars in the k closest annotated frames (see Section 4 for pseudocode). This will produce an estimate of the number of cars in a given unannotated frame. These scores can then be used by BlazeIt's query processing algorithm [31].

Approximate selection. To estimate the probability of matching the predicate (i.e., query-specific proxy score) for SUPG, TASTI will compute the weighted average as above, except that annotated frames that contain a car receive a score of 1 and annotated frames that do not contain a car receive a score of 0. These scores can then be used by SUPG's selection algorithm [33].

We now discuss TASTI's index construction method (Section 3) and TASTI's query processing method (Section 4).

3 INDEX CONSTRUCTION

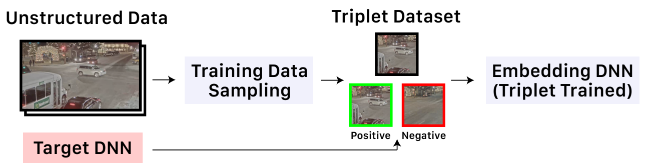
We describe how TASTI constructs indexes, which can be used to produce high quality proxy scores without the use of query-specific proxy models. Many queries only require a low dimensional representation of data records to answer, such as object types and positions (as opposed to raw pixels in a video). Furthermore, in many applications, this low dimensional representation has a natural closeness function, which can be directly used to construct high quality proxy scores. TASTI attempts to construct representations that reflect these heuristics by grouping close records and separating far records.

We show a schematic of the training in Figure 1a, the index construction in Figure 1b, and the overall algorithm in Algorithm 1. TASTI's index construction procedure consists of optionally training an embedding DNN via the triplet loss, producing embeddings per record, selecting cluster representatives, and computing statistics over the cluster representatives.

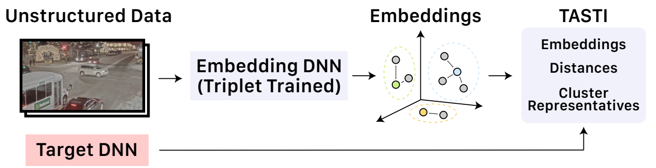
Throughout, we use the furthest point first (FPF) clustering algorithm [19]. FPF iteratively chooses the furthest point from the existing cluster representatives as the new representative. FPF performs well in practice, is computationally efficient, and provides a 2-approximation to the optimal maximum intra-cluster distance (which our analysis uses).

3.1 Training the Embeddings

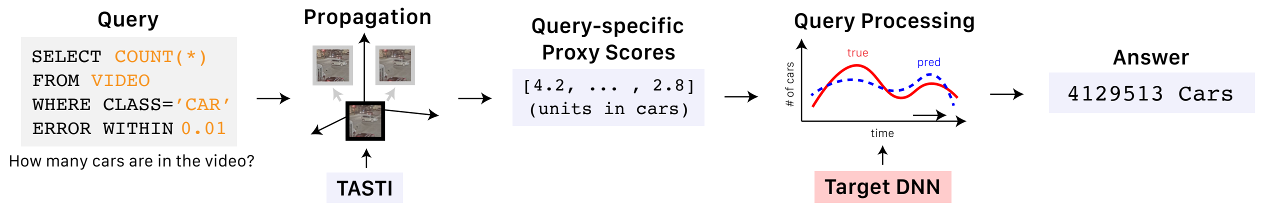
TASTI optionally trains a mapping between data records (e.g., frames of a video) and semantic embeddings. The semantic embeddings have the desideratum that data records that have similar extracted attributes are close in embedding space, and vice versa for records that have dissimilar extracted attributes. For example,



(a) Overview of TASTI's procedure for training embeddings. Training data is selected via the induced schema and a pre-trained embedding DNN. This data is then used to train an embedding DNN via the closeness function provided by the user.



(b) Overview of TASTI's index construction procedure. TASTI computes per-record embeddings, selects sample records to annotate (cluster representatives), and computes embedding distances from the unannotated records to the cluster representatives. TASTI then stores the target labeler results, distances, and embeddings as its index.



```
SELECT COUNT(*)
FROM VIDEO
WHERE CLASS='CAR'
ERROR WITHIN 0.01
```

(c) Overview of TASTI's query processing. Given a query, TASTI will compute exact results on the cluster representatives. It will then produce proxy scores on unannotated records by propagating the exact scores via embedding distance. These scores can then be used in existing downstream query processing algorithms based on proxy scores.

Figure 1: TASTI system overview.

consider queries over object type and position. A frame with a single car in the upper left should be close to another frame with a single car in the upper left, but far from a frame with two cars in the bottom right.

We describe our training method via domain-specific triplet losses and show a schematic in Figure 1a. We note that TASTI's training procedure is optional: pre-trained embeddings can be also be used for the index if training is expensive.

Domain-specific triplet loss. To train the embedding DNN, TASTI uses the triplet loss [42]. The triplet loss takes an anchor point, a positive example (i.e., a close example), and a negative example (i.e., a far example). It penalizes examples where the anchor point and the positive point are further apart than the anchor point and the negative point (see Section 5).

A key choice in using the triplet loss is selecting points that are "close" and those that are "far." This choice is application specific, but many applications have natural choices. For example, any frame of video with different numbers of objects may be far (see Section 2 for pseudocode). Furthermore, frames with the same number of objects, but where the objects are far apart may also considered far.

Training data selection (FPF mining). Training via the triplet loss requires invocations of the target labeler to determine whether pairs of records are close or not. Due to the cost of the target labeler, TASTI must sample records to be selected for training; we assume the user provides a budget of target labeler invocations. While TASTI could randomly sample data points, randomly sampled points may mostly sample redundant records (e.g., majority of empty frames) and miss rare events. We empirically show that randomly sampling training data results in embeddings that perform well on average, but can perform poorly on rare events (Section 6).

Algorithm 1 Pseudocode for TASTI's index construction procedure. Given a dataset X, training points N_1, number of cluster representatives N_2, and min-k to retain, TASTI will construct the index as follows. FPF is the furthest point first algorithm, where the arguments are the embeddings and the number of points to select.

```
function MAKE TASTI INDEX($X$, $N_1$, $N_2$, $k$)
    PretrainedEmbeddings[$i$] ← PretrainedModel($X[i]$)
    TrainingPoints ← FPF(PretrainedEmbeddings, $N_1$)
    TripletModel ← Finetune(TrainingPoints, PretrainedModel)
    Embeddings[$i$] ← TripletModel($X[i]$)
    ClusterRepresentatives ← FPF(Embeddings, $N_2$)
    MinKDistances[i] ← ClosestKDistances($X[i]$, ClusterRepresentatives, $k$)
    return ClusterRepresentatives, MinKDistances
```

To produce embeddings that perform well across queries, we would ideally sample a diverse set of data records. For example, suppose 80% of a video were empty: selecting frames at random would mostly sample empty frames. Selecting frames with a variety of car numbers and positions would be more sample efficient.

When available, TASTI uses a pre-trained DNN to select such diverse points. These pre-trained DNNs are widely available, e.g., DNNs pre-trained on ImageNet [24] or on large text corpora (BERT) [14]. Pre-trained DNNs produce embeddings that are semantically meaningful, although not adapted to the specific induced schema.

To produce training data that results in embeddings that perform well on rare events, TASTI performs the following selection procedure. First, TASTI uses a pre-trained DNN to generate embeddings over the data records. Then, TASTI executes the FPF algorithm to select the training data. TASTI constructs triplets from the training data via target labeler annotations. TASTI will first bucket records by the closeness function. To construct a triplet, TASTI will sample two different buckets at random: it will select the anchor and positive record at random from the first bucket and a negative record at random from the second bucket.

3.2 Clustering

TASTI produces clusters via the embedding DNN. As we describe in Section 4, TASTI propagates annotations/scores from cluster representatives to unannotated data records.

A key choice is which data records to select as cluster representatives. Similar to selecting training data, TASTI could select a set of cluster representatives at random. While random sampling may do well on average at query time, it may perform poorly on rare events (i.e., outliers).

To address this issue, TASTI selects cluster representatives via FPF. FPF chooses points that are far apart in embedding space. Thus, if the embeddings are semantically meaningful, then FPF will select data records that are diverse. Finally, we mix a small fraction of random clusters, which helps "average-case performance" queries.

TASTI stores the distances of all embeddings to each cluster representative. As we describe in Section 4, TASTI uses the k nearest cluster representatives for query processing.

3.3 Cracking TASTI Indexes

In contrast to prior work, which can only share work between queries in an ad-hoc manner, TASTI's proxy scores will improve as queries are executed. In particular, when any query executes the target labeler on a data record, TASTI can cache the target labeler result. The records over which the target labeler are executed can then be added as new cluster representatives. Computing the distance to the new cluster representative is computationally efficient and trivially parallelizable. We note that this is a form of "cracking" [27].

3.4 Computational Performance

Suppose there are N data records, D dimensions, L training iterations, and a total target labeler budget of C. Denote the costs of the target labeler, embedding DNN, and distance computation as c_T, c_E, and c_D respectively. The total cost of index construction is $O(C \cdot c_T + L \cdot c_E + N \cdot c_E + NCD \cdot c_D)$ assuming the cost of a training iteration is proportional to the cost of the forward pass [30].

The ratio of these steps depends on the relative computational costs. In many applications, the cost of embedding is less expensive than the cost of the target labeler. For example, Mask R-CNN can execute as slow as at 3 fps, compared to an embedding DNN which executes at 12,000 fps [35]. Furthermore, human labelers are orders of magnitude more expensive than embedding DNNs (up to 100,000× more expensive).

4 QUERY PROCESSING WITH TASTI

How can TASTI indexes accelerate query processing? We propose automatic methods of construct query-specific proxy scores with TASTI, which can then be passed to existing proxy score-based algorithms. These query-specific proxy scores are an approximation of the result of executing the target labeler on the data records for the particular query. Consider an aggregation query counting the average number of cars per frame [31]. The query-specific proxy scores would be an estimate of the number of cars in a given frame.

Many downstream query processing algorithms only require proxy scores and the target labeler. For example, selection without guarantees (e.g., binary detection) [3, 32, 38], selection with statistical guarantees [33], aggregation [31], and limit queries [31] only require query-specific proxy scores and the target labeler.

TASTI provides a default method of taking the target labeler output and producing a numeric score, which can support aggregation, selection and limit queries. Its default method produces exact scores on the cluster representatives and propagates using the distance-weighted average for numeric columns and distance-weighted majority vote for categorical columns. If desired, a developer may also implement custom functions to produce proxy scores for other queries. We describe several examples of how proxy scores can be computed and used for common query types, and then describe the interface for implementing custom proxy scores. We show a schematic of the query processing procedure in Figure 1c.

4.1 Query Processing Examples

We provide examples of the query-specific scoring functions, score propagation, and downstream query processing for several classes of queries below.

Approximate aggregation. Consider the example of counting the average number of cars per frame, as studied by BlazeIt [31]. The scoring function would take the detected boxes in a frame and return the count of the boxes matching "car," as shown above. For $k = 1$, the query-specific proxy score would be the count for the nearest cluster representative and for $k > 1$, it would be the distanced-weighted mean count of the nearest k cluster representatives for a given frame.

The query-specific proxy scores can be used to answer the query with statistical error bounds, e.g., used as a control variate by the BlazeIt's query processing algorithm. The scores could also be used to directly answer the query.

Selection. Consider a query that selects all frames of a video with a car, as studied by prior work [3, 32, 33, 38]. The scoring function would take the detected boxes in a frame and return 0 if there are no cars and 1 if there is a car in the frame. The query-specific proxy score can be smoothed for $k > 1$.

The query-specific proxy scores can be used as input to SUPG, in which sampling is used to achieve statistical guarantees on the recall or precision of selected records [33]. These scores can also be used directly to answer the query (i.e., return the records with value above some threshold, either ad-hoc or computed over some validation set), as other systems do [3, 32, 38].

Limit queries. Consider a query that selects 10 frames containing at least 5 cars [31]. Such queries are often used to manually study rare events. In this case, the scoring function and query-specific proxy scores would be the same as for aggregation. For limit queries, we generally recommend using $k = 1$, since this query is typically focused on ranking rare events. The query processing algorithm will examine frames with the target labeler as ordered by the query-specific proxy scores. The algorithm will terminate once the requested number of frames is found.

4.2 Custom Proxy Scores

TASTI has built-in functionality to compute and propagate scores for selection, aggregation and limit queries using the methods described in the previous section. In addition, developers may specify custom scores to extend TASTI to supporting other queries.

The API for specifying scoring functions is as follows. Denote the type of the output of the target labeler as `TargetLabelerutput` (e.g., a list of bounding boxes) and the type of the score as `ScoreType` (e.g., a float). Using Python typing, the developer would implement:

```
def Score(target_output: TargetLabelerOutput) -> ScoreType
```

These functions can be implemented in few lines of code. We show the pseudocode for the example above:

```
def CountCarScore(boxes: Sequence[Boxes]) -> int:
    return len([box for box in boxes
                if box.object_type == 'car'])
```

Other queries, e.g., over object positions, can be implemented similarly with few lines of code.

4.3 Score Propagation

Given the query-specific scoring functions, TASTI will execute the scoring functions on the cluster representatives (as the target labeler outputs are available for these data records). In order to execute downstream query processing, TASTI must also materialize approximate scores for the remainder of the data records.

To produce these query-specific proxy scores, TASTI will propagate scores from the cluster representatives to the unannotated records. The score for each data record will be the inverse distance-weighted mean of the nearest k cluster representatives for numeric scores. For categorical scores, TASTI will take the distance-weighted majority vote. Since the distances to cluster representatives are cached, this process is computationally efficient. A developer may also implement a custom method of propagating scores. We show an example of such a method in Section 6.3 for limit queries.

5 THEORETICAL ANALYSIS

We present a statistical performance analysis of our methods to better understand resulting query quality. Intuitively, if the original data records have a metric structure and the triplet loss recovers this structure, we expect downstream queries to behave well. Specifically, we provide guarantees on query quality (typically accuracy) when using TASTI directly. We show that accuracy for a natural class of "smooth" queries is directly connected to the triplet loss and the density of clustering. While the assumptions in our analysis may not hold in practice, we conduct our analysis to provide statistically grounded intuition for why TASTI can outperform baseline methods. We validate our intuition with extensive experiments (Section 6).

We formalize this intuition by analyzing how downstream queries behave under the triplet loss. We specifically analyze the case where $k = 1$, i.e., using a single cluster representative in query processing.

5.1 Notation and Preliminaries

Notation. We define the set of data records as $\mathcal{D} := \{x_1, ..., x_N\}$, the scoring function $f(x_i) : \mathcal{D} \to \mathbb{R}$, and the embedding function $\phi(x_i) : \mathcal{D} \to \mathbb{R}^d$. Denote the cluster representatives as $R := \{x_r : r \in \mathcal{R}\} \subset \mathcal{D}$ for some set $\mathcal{R} \subset \{1, ..., N\}$. Given this set, we denote the representative mapping function as $c(x_i) : \mathcal{D} \to R$, which maps a data record to the nearest cluster representative, and the query-specific scores as $\hat{f}(x) := f(c(x))$.

Suppose there is a query-specific loss function $\ell_Q(x_i, y_i) : \mathcal{D} \times \mathbb{R} \to \mathbb{R}$ where $y_i \in \mathbb{R}$ is the predicted label. ℓ_Q will be used to evaluate the quality of f and $\hat{f}$ as $\ell_Q(x, f(x))$ and $\ell_Q(x, \hat{f}(x))$.

We define the per-example triplet loss as

$$\ell_T(x_a, x_p, x_n; \phi, m) := \max(0, m + |\phi(x_a) - \phi(x_p)| - |\phi(x_a) - \phi(x_n)|)$$

where we omit ϕ and m where clear. Define the ball of radius M as $B_M(x) = \{x' : d(x, x') < M\}$ and its complement $\bar{B}_M$. For random variables $x_a \sim \mathcal{D}$, $x_p \sim B_M(x_a)$, and $x_n \sim \bar{B}_M(x_a)$ drawn uniformly from the sets, we define the population triplet loss as

$$L(\phi; M, m) := \mathbb{E}_{x_a, x_p, x_n}[\ell_T(x_a, x_p, x_n; \phi, m)] \quad (1)$$

for some margin $m > 0$.

Assumptions and properties. We make the following assumptions. We first assume that there is a metric $d(x_i, x_j)$ on $\mathcal{D}$ and that $\mathcal{D}$ is compact with metric d. We further assume that $\ell_Q(x, y)$ is Lipshitz in x and y with constant $K_Q/2$, in both arguments.

For both of our proofs, we assume the triplet loss is low and the cluster representatives are dense enough under ϕ. Low triplet loss controls the quality of the embeddings with respect to the original metric d. The density of the cluster representatives controls how close the unannotated records are from the cluster representatives in the original space.

Example. Consider the video setting described in Section 2. $\mathcal{D}$ is the set of frames, ϕ is the trained embedding DNN, and we use the metric induced by closeness function also described in Section 2. Consider the two queries: aggregation queries for the number of cars and selecting frames of cars. For the aggregation query, f maps frames to the number of cars. For the selection query, f maps frames with cars to 1 and frames without cars to 0.

5.2 Theorem Statements

We defer all proofs to Appendix A.

Zero loss case. To theoretically analyze our index and query processing algorithms, we first consider the case where the embedding achieves zero triplet loss (we generalize to non-zero loss below). We show the following positive result: using the query-specific proxy scores in this setting will achieve bounded loss. In fact, for ℓ_Q that are identically 0 (e.g., for the example above), TASTI will achieve *exact* results.

We now state the main theorem for the zero-loss case.

THEOREM 1 (ZERO LOSS). *Let ϕ be an embedding that achieves $L(\phi; M, m) = 0$ and c be such that $\max_{x \in \mathcal{D}} |\phi(x) - \phi(c(x))| < m$. Then, the query procedure will suffer an expected loss gap of at most*

$$\mathbb{E}[\ell_Q(x, \hat{f}(x))] \leq \mathbb{E}[\ell_Q(x, f(x))] + M \cdot K_Q. \quad (2)$$

Generalization to Non-zero Loss. We generalize our analysis to the non-zero loss case below. We show that the loss in queries is bounded by the triplet loss and several other natural quantities.

THEOREM 2 (NON-ZERO LOSS). *Consider an embedding ϕ that achieves $L(\phi; M, m) = \alpha$ and a clustering c such that $\max_{x \in \mathcal{D}} |\phi(x) - \phi(c(x))| < m$. Assume that the query loss ℓ_Q is upper bounded by C. Then, query procedure will suffer an expected loss gap of at most*

$$\mathbb{E}[\ell_Q(x, \hat{f}(x))] \leq \mathbb{E}[\ell_Q(x, f(x))] + M \cdot K_Q + \frac{C \sup_x |\bar{B}_M(x)|}{m} \alpha. \tag{3}$$

5.3 Discussion

We have shown that many classes of queries will have bounded loss (i.e., discrepancy from exact answers). However, we note that our analysis has several limitations. First, TASTI uses the nearest $k = 5$ cluster representatives to generate the query-specific proxy scores by default, not $k = 1$ as used in our analysis. Second, the triplet loss may be large in practice. Third, not all queries admit Lipschitz losses. Nonetheless, we believe our analysis provides intuition for why TASTI outperforms even recent state-of-the-art. We defer a more detailed analysis to future work.

6 EVALUATION

We evaluated TASTI on five real world datasets using three query types. We describe the experimental setup and baselines. We then demonstrate that TASTI's index construction is cheaper than recent state-of-the-art executed end-to-end, that TASTI's proxy scores outperforms per-query proxies on all settings we consider, that all components of TASTI are required for performance, and that TASTI is not sensitive to hyperparameter settings. Our anonymized code is available at https://anonymous.4open.science/r/tasti-76FA.

6.1 Experimental Setup

Datasets, target labelers, and triplet loss. We considered three video datasets, a text dataset, and a speech dataset. We used the `night-street`, `taipei`, and `amsterdam` videos as used by BLAZEIT [31]. The `night-street` dataset is widely used in video analytics evaluations [11, 31, 32, 43]. The `taipei` dataset has two object classes (car and bus) and we use the *same* set of embeddings for both. We used Mask R-CNN as the target labeler and ResNet-18 as our embedding DNN. The closeness function separates frames with objects that are far apart and frames with different numbers of objects (when also considering object types).

For the text dataset, we used a semantic parsing dataset [46]. The dataset consists of pairs of natural language questions and corresponding SQL statements. We assumed the SQL statements are not known at query time and must be annotated by crowd workers (i.e., that crowd workers are the target labeler). We used BERT [14] for the embedding DNN. We considered queries over SQL operators and number of predicates. The closeness function separates questions over different SQL operators and number of predicates.

For the speech dataset, we used the Common Voice dataset [4]. The dataset consists of short speech snippets. We assumed that the attributes of speaker gender and age are not known at query time and must be annotated by crowd workers. We used an audio ResNet-22 [36] for the embedding DNN. The closeness function separates records by gender and discretized age bucket.

Queries and metrics. We evaluated TASTI and per-query proxies on three classes of queries using three recently proposed algorithms: aggregation, selection, and limit queries.

Our primary cost metric across all queries is the number of target labeler invocations and also report end-to-end costs for certain experiments. We use target labeler invocations as the primary metric for several reasons. First, in many cases, the target labeler is actually a human labeler, particularly when used in social or life sciences [34]. Second, the target labelers we evaluate are thousands to hundreds of times more expensive than query processing costs and proxy models [35], and thus make up the majority of query costs. In addition, this strictly benefits systems that use per-query proxies which must be executed at query time: TASTI does not train a model per query.

Aggregation. For aggregation queries, we queried for an approximate statistic of the target labeler executed on the unstructured data records. We computed the average number of objects per frame for the video datasets, the average number of predicates per query for the WikiSQL dataset, and the fraction of male speakers in the Common Voice dataset.

For all settings, we used the EBS sampling as used by the BLAZEIT system [31], which provides guarantees on error. EBS sampling uses the proxy scores to guide target labeler sampling. Better proxy scores will result in fewer target labeler invocations. As such, we measured the number of target labeler invocations (lower is better).

We additionally compare TASTI to approximate aggregation without statistical guarantees, which uses the proxy scores to answer queries directly (as used by BLAZEIT).

Selection. For selection queries, we executed approximate selection queries with recall targets (SUPG queries [33]). We selected for frames with objects for video datasets, natural language questions that are parsed into selection SQL statements for the WikiSQL dataset, and male speakers in the Common Voice dataset.

Given a target labeler budget, these queries return a set of records matching a predicate with a given recall target with a given confidence level (e.g., "return 90% of instances of cars with 95% probability of success"): these queries are useful in scientific applications or mission-critical settings [33]. In contrast to queries that do not provide statistical guarantees, SUPG guarantees the recall target with high probability. Since recall SUPG queries fix the number of target labeler invocations, we measured the false positive rate (lower is better).

We additionally compare TASTI to approximate selection without statistical guarantees, which uses the proxy scores to answer queries directly. We slightly modify the query processing algorithms of NOSCOPE, Tahoma, and probabilistic predicates to directly use proxy scores and use the accuracy metric of F1 score.

Limit. For limit queries, we used the ranking algorithm proposed by Kang et al. [31]. This ranking algorithm examines data records that are likely to match the predicate of interest in descending order by the proxy score. Proxy scores that have high recall for given number of records will perform better. As such, we measured the number of target labeler invocations (lower is better).

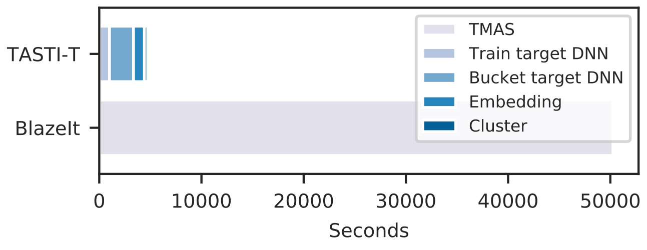


Figure 2: Breakdown of time to construct indexes for TASTI and for BlazeIt on the `night-street` dataset. The BlazeIt index is the "target-model annotated set" (TMAS) [31]. Similar results hold for other datasets.

Methods evaluated. We used the query processing methods above and use the per-query proxies as used in Kang et al. [31] (aggregation and limit queries) and Kang et al. [33] (selection queries). We use the exact proxy models for the video datasets (a "tiny ResNet"), logistic regression over FastText embeddings [9] for the WikiSQL dataset, and a smaller CNN (CNN-10) [36] for the Common Voice dataset. FastText embeddings are less expensive than BERT embeddings.

Throughout, we refer to TASTI when using a pre-trained DNN as the embedding DNN as "TASTI-PT" (pre-trained) and TASTI when using a triplet-loss trained embedding DNN as "TASTI-T" (trained). We demonstrate that TASTI-T generally outperforms TASTI-PT.

Hardware and timing. We evaluated TASTI on a private server with a single NVIDIA V100 GPU, 2 Intel Xeon Gold 6132 CPUs (56 hyperthreads), and 504GB of memory. In contrast to prior work, we timed end-to-end query processing times for TASTI, *including* the video loading and embedding DNN execution times, which is excluded in prior work [31].

Due to the large cost of executing the target labeler, we simulated its execution by caching target labeler results and computing the average execution time for the target labeler. For baselines, we only timed the target labeler computation and exclude the computational cost of proxy models, which strictly improves the baselines. We excluded the cost of query processing [31, 33] as it is negligible in all cases. Namely, the query processing is over orders of magnitude less expensive than target labeler invocation for all queries we consider.

6.2 Index Construction Performance

To understand the index construction performance, we measured the wall clock time to construct TASTI indexes. We compared to BlazeIt, which constructs indexes by executing the target labeler on a subset of the data (referred to as the "TMAS" [31]). For BlazeIt, we only considered the cost of constructing the TMAS. For TASTI, we measured the full index construction time, including the embedding DNN training and distance computation times. We computed the construction times on the `night-street` dataset; similar results hold for other datasets.

We show the breakdown of index construction time for TASTI and BlazeIt in Figure 2 using the parameters in Section 6.3. TASTI requires far fewer target labeler invocations for index construction, so is substantially faster than BlazeIt. We additionally show the index construction time vs performance for BlazeIt and a range of parameters for TASTI (Figure 3). TASTI can outperform or match BlazeIt performance with up to 10× less expensive index construction times.

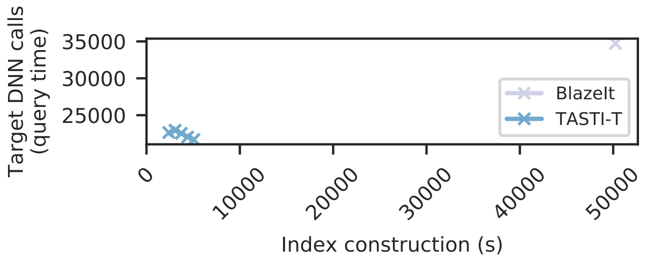


Figure 3: Index construction time vs performance of TASTI and BlazeIt for aggregation queries on the `night-street` dataset. Similar results hold for other datasets.

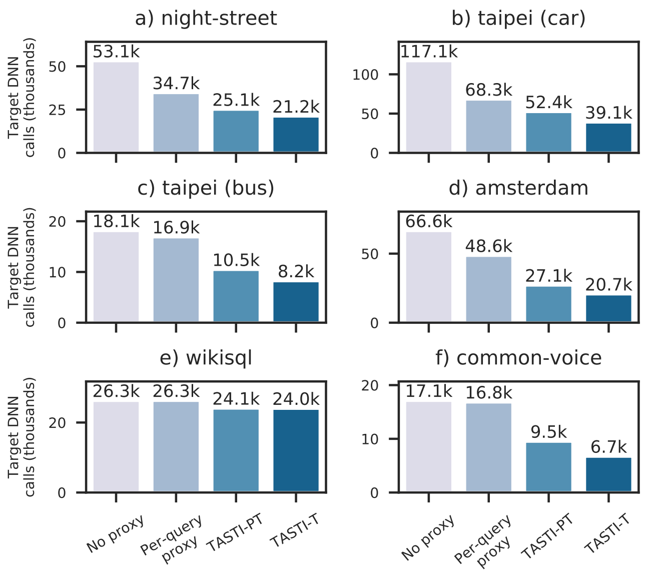


Figure 4: Number of target labeler invocations for baselines and TASTI for approximate aggregation queries (lower is better). As shown, TASTI outperforms baselines in all cases, including prior, per-query proxy state-of-the-art by up to 2×. All methods achieved the target accuracy.

6.3 End-to-end Performance

We show that TASTI outperforms recent state-of-the-art per-query proxy methods for approximate aggregation, selection with guarantees, and limit queries. For all video datasets in this section, we used 3,000 training records, 7,000 cluster representatives, and an embedding size of 128. To show the generality of TASTI, we used a single set of embeddings/distances for both `taipei` classes. For the WikiSQL and Common Voice datasets, we used 500 training examples and 500 cluster representatives. We measured the query processing costs or query accuracy in this section.

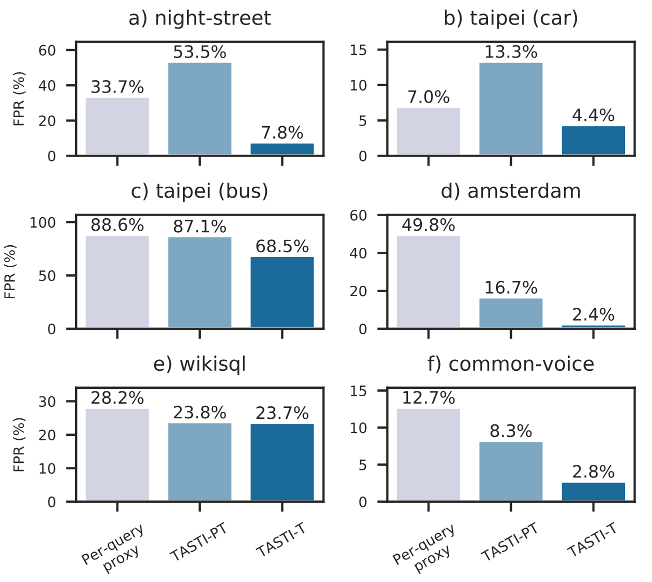


Figure 5: False positive rate for recall-target SUPG queries (lower is better). We show the performance of baselines and TASTI. As shown, TASTI outperforms baselines in all cases.

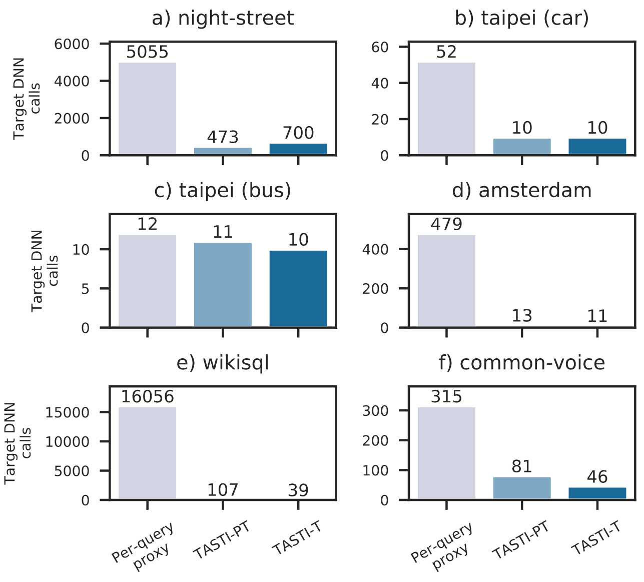


Figure 6: Number of target labeler invocations for baselines and TASTI for limit queries (lower is better). TASTI outperforms baselines in all cases, including prior state-of-the-art by up to 34×.

Approximate aggregation. For approximate aggregation queries, we compared TASTI to using no proxy (random sampling) and an ad-hoc trained proxy model. We used the exact experimental setup as BlazeIt [31] for video datasets, which targeted an error of 0.01 and a success probability of 95%. We aggregated over the average number of objects per frame for all video datasets (cars or buses), the number of clauses per statement in the WikiSQL dataset, and the fraction of male speakers in the Common Voice dataset.

As shown in Figure 4, TASTI outperforms for aggregation queries on all datasets. In particular, TASTI outperform state-of-the-art per-query proxies for aggregation queries (BlazeIt) by up to 2× with less expensive index construction costs. Further, TASTI outperforms no proxy by up to 3×.

TASTI's improved performance comes from better query-specific proxy scores (ρ^2 of 0.91 vs 0.55). As the correlation of the proxy scores with the target labeler increases, the control variates variance decreases. Reduced variance results in fewer samples, as the EBS stopping algorithm is adaptive with the variance.

Selection. For selection queries with statistical guarantees (SUPG queries), we compared TASTI to using an ad-hoc trained proxy model (standard random sampling is not appropriate for SUPG queries). We used the exact same experimental setup as in SUPG [33] for the video datasets. For all queries, we used a recall target of 90% with a confidence of 95%, as used in [33]. We search for cars or buses in the video datasets, star operators for WikiSQL, and male speakers in the Common Voice dataset.

As shown in Figure 5, TASTI outperforms on all datasets. In particular, TASTI can improve the false positive rate by almost 21× over recent state-of-the-art. We further show that the triplet training

Target	TASTI (no index)	TASTI (all costs)	Uniform (no proxy)	Exhaustive
Human labeler	**$1,482**	$1,972	$3,717	$68,116
Mask R-CNN	**7,060 s**	9,474 s	17,702 s	324,362 s
SSD	**141 s**	269 s	354 s	6,487 s

Table 1: Query costs for TASTI (when amortizing the cost of constructing the index), TASTI (including the cost of the index), uniform sampling, and exhaustive labeling for answering an approximate aggregation query on the `night-street` dataset. TASTI, both with and without the indexing costs, outperforms in all cases.

improves performance. As with aggregation queries, TASTI's improved performance comes from better query-specific proxy scores (ρ^2 of 0.90 vs 0.79).

Limit queries. For limit queries, we used the ranking algorithm proposed by BlazeIt [31]. We use the exact same experimental setup as BlazeIt for the video datasets (including the query configurations, e.g., number of objects, etc.). For limit queries, we use a custom scoring function which is the regular scoring function with $k = 1$ and ties broken by distance to the cluster representatives.

Figure 6 shows TASTI outperforms on all datasets. TASTI can improve performance by up to 24× compared to recent state-of-the-art. As we demonstrate, TASTI's FPF mining and FPF clustering are critical for performance when searching for rare events (Section 6.7, Figure 9 and 10). The FPF algorithm naturally produces clusters that are far apart, which is beneficial when searching for rare events.

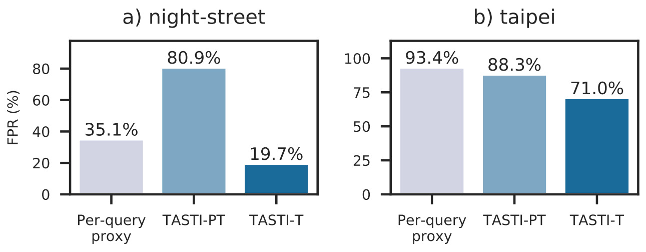


Figure 7: SUPG queries for selecting objects of interest on the left hand side of the frame. This query violates the Lipschitz condition, but TASTI still outperforms baselines.

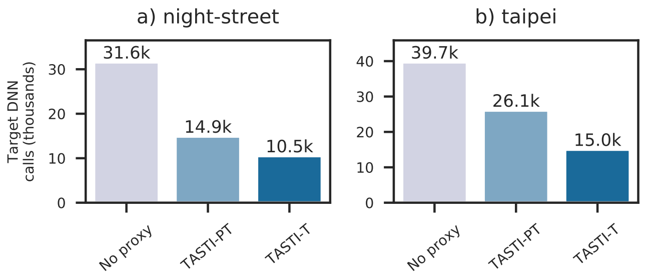


Figure 8: Aggregation query for the average x position of objects in frames of a video. Recent state-of-the-art is not well suited for this query as regression can be difficult for proxy models. In contrast, TASTI performs well on these queries.

Comparison without proxies (aggregation). To illustrate the performance of TASTI, we also compare total costs of uniform sampling with no index and to exhaustive labeling. In these experiments, we compare against three different target labelers: human labelers, Mask R-CNN, and SSD (an inexpensive object detection method). Different applications require different levels of accuracy: the human labeler is the most accurate, followed by Mask R-CNN, and finally SSD. Importantly, we note that SSD is over 2× less accurate than Mask R-CNN (50.2 vs 23.0 mAP), so can result in inaccurate queries when used as the target labeler.

We show the cost of TASTI (index cost amortized), TASTI (including index cost), uniform sampling, and exhaustive labeling in Table 1 for the three different targets for aggregation on the `night-street` dataset. We use the same approximate aggregation query as above, targeting an error of 0.01 and a success probability of 95%. As shown, the cost when using TASTI can be up to 46× cheaper to answer aggregation queries. Importantly, TASTI is cheaper in all cases *when including the cost of building the index* for answering this query.

Finally, we note that cheaper models are less accurate: SSD results in a 33% error compared to the more accurate Mask R-CNN. Matching the accuracy of TASTI with Mask R-CNN as the target labeler takes 30.4 s to execute, which is cheaper than exhaustively executing SSD.

6.4 New Queries

In addition to the queries above, we demonstrated that TASTI can be used to efficiently answer queries that prior work is not well suited for. We considered two queries over positions of objects in video. In particular, these tasks require modified data preprocessing or losses for proxy models, but TASTI can naturally produce proxy scores for both tasks.

Selecting objects by position. We considered the query of selecting objects in the left hand side of the video, as measured by the average x-position of the bounding box. We compared TASTI to to training a proxy model by extending SUPG and to TASTI without triplet training. Results are shown in Figure 7.

Prior proxy models were not designed to take position into account, which may explain their poor performance: there is a sharp discontinuity for labels in the center of the frame. Learning the boundary in the frame may require large amounts of training data. In contrast, TASTI performs well on as it uses the information from the target labeler, despite the query violating our assumptions in the theoretical analysis. As shown, TASTI outperforms both baselines.

Average position. We consider the query of computing the average position of objects in frames of video (specifically the x-coordinate). We compare TASTI to random sampling (no proxy) and to TASTI without triplet training. We attempted to train a BlazeIt proxy model by regressing the output to the average position but were unable to train a model that outperformed random sampling. BlazeIt was not configured for such queries and that we are unaware of work on proxy models for pure regression. We show results in Figure 8. As shown, TASTI outperforms random sampling by up to 3×, without having to implement custom training code for a new proxy model.

Dataset	Method	Query	Quality metric
`night-street`	TASTI	Agg.	**3.3%**
`night-street`	BlazeIt	Agg.	4.4%
`night-street`	TASTI	Selection	**5.5**
`night-street`	NoScope	Selection	14.9

Table 2: Performance of TASTI and baselines on queries without statistical guarantees (lower is better). The quality metrics are percent error and 100 - F1 score for aggregation and selection queries respectively. TASTI outperforms on all settings we considered.

6.5 Queries Without Guarantees

In addition to queries with statistical guarantees, we executed aggregation and selection queries without statistical guarantees. For aggregation queries, we used the proxy score to directly compute the statistic of interest and measured the percent error from the ground truth. For selection queries, we used the proxy score to select records above some threshold. As some selection queries are class imbalanced, we measured 100 - F1 score (so lower is better).

We show results for TASTI and for proxy model-based baselines in Table 2. As shown, TASTI outperforms on quality metrics for all settings we considered, indicating that TASTI's proxy scores are higher quality.

Dataset	1st query	2nd query	Quality metric
`night-street`	Agg.	SUPG	4.9% (8.6%)
`taipei`	Agg.	SUPG	40.1% (55.9%)
`night-street`	SUPG	Agg.	18.9k (21.2k)
`taipei`	SUPG	Agg.	34.6k (39.1k)

Table 3: Performance of TASTI after cracking. We measured query performance of a SUPG/aggregation query after cracking (false positive rate and number of target labeler invocations, lower is better for both). Results after cracking are shown along with results before cracking in parentheses. TASTI improves results in all settings we tested.

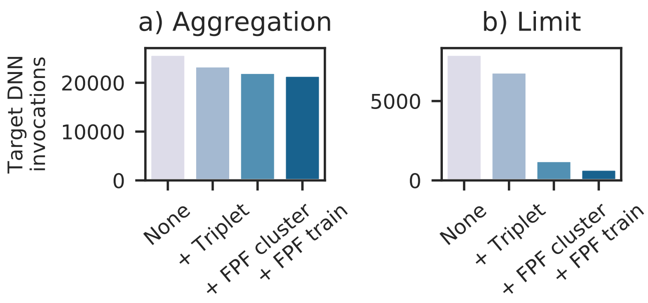


Figure 9: Factor analysis, in which optimizations are added in in sequence. As shown, all optimizations improve performance for aggregation queries. For limit queries, FPF training and clustering are required for triplet training to improve performance.

6.6 Cracking

We further demonstrated that TASTI's indexes can be "cracked" [27]. To show this, we executed an aggregation query followed by a SUPG query and vice versa. We used the target labeler annotations from the first query to improve TASTI's index before executing the second query. We use the same quality/runtime metrics as for queries with statistical guarantees.

As shown in Table 3, TASTI improves in performance for both queries. In particular, TASTI can improve (decrease) the false positive rate for SUPG queries by up to 1.7× after repeated queries.

6.7 Factor Analysis and Lesion Study

We investigated whether all of TASTI's components contributes to performance. We find that all components of TASTI (triplet loss, FPF mining, and FPF clustering) are critical to performance.

Factor analysis. We first performed a factor analysis, in which we began with no optimizations and added the triplet loss, FPF mining, and FPF clustering in turn. For brevity, we show results for the `night-street` dataset for aggregation and limit queries. Aggregation queries highlight "average-case" performance and limit queries highlight "rare-event" performance. We choose the `night-street` dataset as it has been widely studied in visual analytics [11, 31–33, 43]; other datasets have similar behaviors.

As shown in Figure 9, all optimizations help performance. In particular, FPF clustering substantially improves limit query performance, as it selects frames that are semantically distinct.

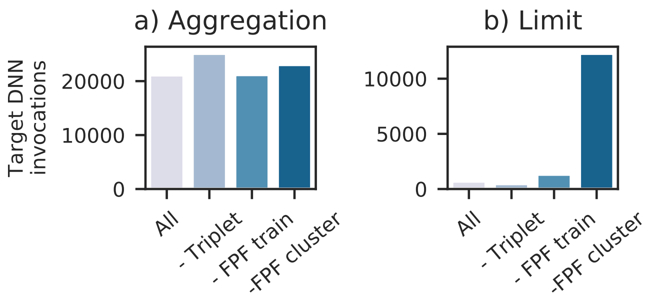


Figure 10: Lesion study, in which optimizations are removed individually (they are not removed cumulatively). As shown, all optimizations improve performance. Lower is better for both aggregation and limit queries.

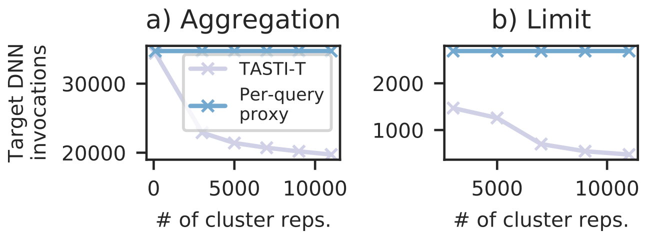


Figure 11: Number of cluster representatives vs performance on aggregation and limit queries on the `night-street` dataset. As shown, TASTI outperforms baselines on a range of parameter settings.

Lesion study. We then performed a lesion study, in which we start with all optimizations, and remove each optimization individually (triplet loss, FPF mining, and FPF clustering). As with the factor analysis, we show results for the `night-street` dataset for aggregation and limit queries; other datasets have similar behaviors.

We show results in Figure 10. As shown, triplet training significantly improves aggregation performance. Furthermore, FPF clustering is critical for limit query performance.

6.8 Sensitivity Analysis

We investigated whether TASTI is sensitive to hyperparameters by varying the number of training examples, number of cluster representatives, and embedding size. As we show, TASTI outperforms baselines on a wide range of parameter settings, demonstrating that hyperparameters are not difficult to select.

Number of buckets. A critical parameter that determines TASTI performance is the number of buckets in the index. To understand the effect of the number of buckets on performance, we vary the number of buckets and measured performance on aggregation and limit queries on the `night-street` dataset. We used 3,000, 5,000, 7,000, 9,000, and 11,000 buckets for both queries. For aggregation queries, we additionally used 50 buckets.

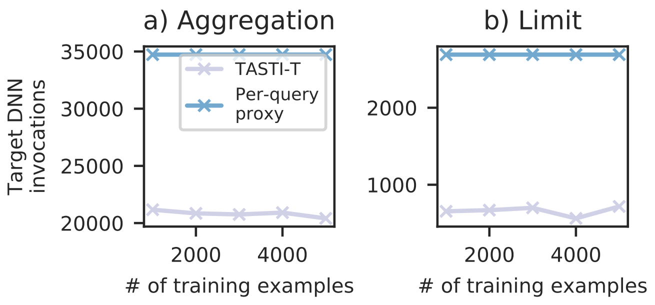


Figure 12: Number of training examples vs performance on aggregation and limit queries on the `night-street` dataset. As shown, TASTI outperforms baselines on a range of parameter settings.

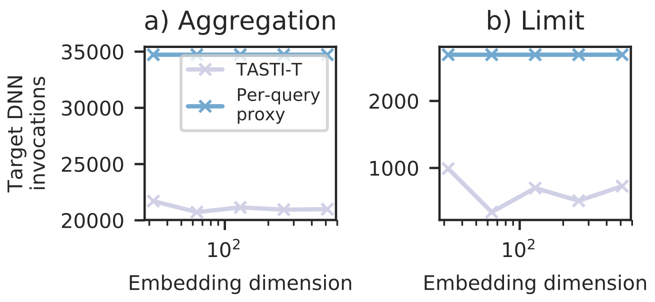


Figure 13: Embedding size vs performance on aggregation and limit queries on the `night-street` dataset. TASTI outperforms baselines on a range of parameter settings.

As shown in Figure 11, TASTI performance improves as the number of buckets increases. For aggregation queries, TASTI outperforms with as few as 50 buckets. For limit queries, TASTI outperforms with 5,000 buckets. We note that this setting still corresponds to index construction over 10× less expensive than the baseline.

Number of training examples. To understand how the number of training examples affects the performance of TASTI, we used 1,000, 2,000, 3,000, 4,000, and 5,000 training examples. We measured performance on aggregation and limit queries on the `night-street` dataset. As shown in Figure 12, the performance of TASTI does not significantly change with the number of training examples. TASTI outperforms baselines across all settings we consider.

Embedding size. To understand how the embedding size affects performance, we varied the embedding size and measured performance on aggregation and limit queries on the `night-street` dataset. We used embedding sizes of 32, 64, 128, 256, and 512. We show results in Figure 13. As shown, TASTI outperforms per-query proxies across a range of embedding sizes for aggregation and limit queries.

7 RELATED WORK

DNN-based queries. Recent work in the database and systems community has focused on accelerating DNN-based queries. Many systems have been developed to accelerate certain classes of queries, including selection without statistical guarantees [3, 26, 32, 38], selection with statistical guarantees [33], aggregation queries, limit queries [31], tracking queries [6], and other queries. These systems reduce the cost of expensive target labeler, often by using cheap, query-specific proxy models. In this work, we propose a general index to accelerate many such queries over the schema induced by the target labeler. We leverage many of the downstream query processing techniques used in this prior work.

Other work assumes that the target labeler is not expensive to execute or that extracting bounding boxes is not expensive [17, 45]. We have found that many applications require accurate and expensive target labelers, so we focus on reducing executing the target labeler wherever possible.

Structured data indexes. There is a long history in the database literature of indexes for structured data [41]. These indexes generally are used to accelerated lookups on certain columns. Techniques range from tree-like structures [8, 13, 25] to hash tables [18]. However, these indexes assume that the data is present in a structured format, which is not the case for the data we consider.

Unstructured data indexes. The analytics community has also long studied indexes for unstructured data. Many of these indexing methods are modality-specific, such as indexes for spatial data [12, 20], time series [2, 15], and low-level visual features [16, 40]. Other indexes accelerate KNN search in possibly high dimensions, when the distances are meaningful [28, 44]. Work in retrieval has used indexed embeddings to accelerate search for semantically similar items, in particular for visual data [5, 37, 45]. While this work also accelerates queries over unstructured data, our work differs in focusing on constructing proxy scores to address unique challenges when queries require executing expensive target labelers.

Coresets. Several communities, including the theory and deep learning communities, have considered coresets [1], which are concise summaries of data. They have been used for nearest neighbor searches [22], streaming data [10], active learning [39], and other applications. We are unaware of work that trains embedding DNNs as an index for proxy score generation.

8 CONCLUSION

To reduce the cost of queries using expensive target labelers, we introduce a method of constructing indexes for unstructured data. TASTI relies on the key property that many queries only require access to target labelers outputs, which are often highly redundant. TASTI uses an embedding DNN and target labeler annotated cluster representatives as its index, which allows for more accurate and generalizable proxy scores across a range of query types. We theoretically analyze TASTI to understand its statistical accuracy. We show that these indexes can be constructed up to 10× more efficiently than recent work. We further show that they can be used to answer queries up to 24× more efficiently than recent state-of-the-art.

A PROOFS OF LEMMAS AND THEOREMS

LEMMA 1. *If $\ell_T(x_a, x_p, x_n) = 0$ for $x_a \in \mathcal{D}, x_p \in B_M(x_a), x_n \in \bar{B}_M(x_a)$, then for all x_i, x_r such that $|\phi(x_i) - \phi(x_r)| < m$ we have $d(x_i, x_r) < M$.*

Proof. To prove the lemma, we will show the contra-positive: if $d(x_i, x_r) \geq M$ implies that $|\phi(x_i) - \phi(x_r)| \geq m$, we have our result.

Let $x_a = x_i, x_n = x_r, x_p \in B_M(x_i)$. $B_M(x_i)$ must be nonempty since $x_i \in B_M(x_i)$. This implies in inequality:

$$0 \geq m + |\phi(x_a) - \phi(x_p)| - |\phi(x_a) - \phi(x_n)|$$
$$|\phi(x_a) - \phi(x_n)| \geq m.$$

□

Proof of Theorem 1. Since the triplet loss is bounded below by 0, $L(\phi; M, m) = 0$ implies that $\ell_T(x_a, x_p, x_n) = 0$ for any x_a, x_n such that $d(x_a, x_n) > M$, since ℓ_T is bounded below by 0. By Lemma 1 with $x_i = x$ and $x_r = c(x)$, and since the maximum intra-cluster instance is m,

$$d(x, c(x)) < M$$

for all $x \in \mathcal{D}$.

Then for every x:

$$\begin{aligned} &|\ell_Q(x, f(x)) - \ell_Q(x, \hat{f}(x))| && (4)\\ &\quad \leq |\ell_Q(x, f(x)) - \ell_Q(c(x), f(c(x)))| + \\ &\quad\quad |\ell_Q(c(x), f(c(x))) - \ell_Q(x, f(c(x)))| && (5)\\ &\quad \leq M \cdot K_Q && (6) \end{aligned}$$

This follows by the definition of $\hat{f}$, the Lipschitz condition of ℓ_Q, and the non-negativity of ℓ_Q.

The proof follows from taking expectations. □

Lemma 2.

$$\mathbb{P}\left[\inf_{x' \in \bar{B}_M(x)} |\phi(x) - \phi(x')| \leq |\phi(x) - \phi(x_p)|\right] \geq \mathbb{P}[d(x, c(x)) > M]$$

for any distribution of x_p such that the condition distribution of x_p on x has support on $B_M(x)$.

Proof. Recall that $c(x) := \arg\min_{x_r \in \mathcal{R}} |\phi(x) - \phi(x_r)|$ and that $d(x, c(x)) > M$ implies $c(x) \in \bar{B}_M(x)$. Then, we have that $\inf_{x' \in \bar{B}_M(x)} |\phi(x) - \phi(x')| \leq |\phi(x) - \phi(x_p)|$ for all $x_p \in B_M(x)$. This gives us the lemma. □

Lemma 3.

$$\frac{1}{m}\ell_T(x, x_p, x_n) \geq \mathbb{1}_{|\phi(x) - \phi(x_n)| \leq |\phi(x) - \phi(x_p)|}$$

for any $x_n \in \bar{B}_M(x), x_p \in B_M(x)$.

Proof.

$$\begin{aligned} \frac{1}{m}\ell_T(x, x_p, x_n) &= \frac{1}{m} \cdot \max(0, m + |\phi(x) - \phi(x_p)| - |\phi(x) - \phi(x_n)|) \\ &= \max\left(0, 1 - \left(\frac{|\phi(x) - \phi(x_n)| - |\phi(x) - \phi(x_p)|}{m}\right)\right) \\ &\geq \mathbb{1}_{|\phi(x) - \phi(x_n)| \leq |\phi(x) - \phi(x_p)|}. \end{aligned}$$

which follows from the hinge dominating the indicator.

□

Proof of Theorem 2. Consider the indicators $\mathbb{1}_{d(x,c(x)) \leq M}$ and its complement $\mathbb{1}_{d(x,c(x)) > M}$.

We analyze

$$\mathbb{E}[\ell_Q(x, \hat{f}(x))] = \mathbb{E}[\ell_Q(x, \hat{f}(x)) \cdot \mathbb{1}_{d(x,c(x)) \leq M}] + \mathbb{E}[\ell_Q(x, \hat{f}(x)) \cdot \mathbb{1}_{d(x,c(x)) > M}]$$

By Theorem 1 and that expectations of indicators are bounded above by 1, we have that

$$\mathbb{E}[\ell_Q(x, \hat{f}(x)) \cdot \mathbb{1}_{d(x,c(x)) \leq M}] \leq \mathbb{E}[\ell_Q(x, f(x))] + M \cdot K_Q$$

To show the RHS we observe that

$$\begin{aligned} &\frac{\sup_x |\bar{B}_M(x)|}{m}\mathbb{E}[\ell_T(x, x_p, x_n)] \\ &\quad \geq \frac{1}{m}\mathbb{E}_{x,x_p}\left[\sum_{x_n' \in \bar{B}_M(x)} \frac{\sup_x |\bar{B}_M(x)|}{|\bar{B}_M(x_n')|}\ell_T(x, x_p, x_n')\right] \\ &\quad \geq \frac{1}{m}\mathbb{E}\left[\sup_{x_n' \in \bar{B}_M(x)} \ell_T(x, x_p, x_n')\right] \\ &\quad \geq \mathbb{E}\left[\inf_{x_n^* \in \bar{B}_M(x)} \frac{1}{m}\ell_T(x, x_p, x_n^*)\right] \\ &\quad \geq \mathbb{E}\left[\mathbb{1}_{\inf_{x' \in \bar{B}_M(x)} |\phi(x) - \phi(x')| \leq |\phi(x) - \phi(x_p)|}\right] \\ &\quad \geq \mathbb{P}[d(x, c(x)) > M] \end{aligned}$$

which follow from Lemmas 2 and 3.

Taking expectations, using Hölder's inequality, and maximizing ℓ_Q gives us the result.

□

REFERENCES

[1] Pankaj K Agarwal, Sariel Har-Peled, Kasturi R Varadarajan, et al. 2005. Geometric approximation via coresets. *Combinatorial and computational geometry* 52 (2005), 1–30.

[2] Rakesh Agrawal, Christos Faloutsos, and Arun Swami. 1993. Efficient similarity search in sequence databases. In *International conference on foundations of data organization and algorithms*. Springer, 69–84.

[3] Michael R Anderson, Michael Cafarella, Thomas F Wenisch, and German Ros. 2019. Predicate Optimization for a Visual Analytics Database. *ICDE* (2019).

[4] Rosana Ardila, Megan Branson, Kelly Davis, Michael Henretty, Michael Kohler, Josh Meyer, Reuben Morais, Lindsay Saunders, Francis M Tyers, and Gregor Weber. 2019. Common voice: A massively-multilingual speech corpus. *arXiv preprint arXiv:1912.06670* (2019).

[5] Artem Babenko, Anton Slesarev, Alexandr Chigorin, and Victor Lempitsky. 2014. Neural codes for image retrieval. In *European conference on computer vision*. Springer, 584–599.

[6] Favyen Bastani, Songtao He, Arjun Balasingam, Karthik Gopalakrishnan, Mohammad Alizadeh, Hari Balakrishnan, Michael Cafarella, Tim Kraska, and Sam Madden. 2020. MIRIS: Fast Object Track Queries in Video. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 1907–1921.

[7] Favyen Bastani, Oscar Moll, and Sam Madden. 2020. Vaas: video analytics at scale. *Proceedings of the VLDB Endowment* 13, 12 (2020), 2877–2880.

[8] Rudolf Bayer and Edward McCreight. 1970. Organization and maintenance of large ordered indexes. In *SIGFIDET Workshop on Data Description, Access and Control*.

[9] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics* 5 (2017), 135–146.

[10] Vladimir Braverman, Dan Feldman, and Harry Lang. 2016. New frameworks for offline and streaming coreset constructions. *arXiv preprint arXiv:1612.00889* (2016).

[11] Christopher Canel, Thomas Kim, Giulio Zhou, Conglong Li, Hyeontaek Lim, David Andersen, Michael Kaminsky, and Subramanya Dulloor. 2019. Scaling Video Analytics on Constrained Edge Nodes. *SysML* (2019).

[12] Paolo Ciaccia, Marco Patella, and Pavel Zezula. 1997. M-tree: An E cient Access Method for Similarity Search in Metric Spaces. In *Proceedings of the 23rd VLDB conference, Athens, Greece*. Citeseer, 426–435.

[13] Douglas Comer. 1979. Ubiquitous B-tree. *ACM Computing Surveys (CSUR)* 11, 2 (1979), 121–137.

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[15] Christos Faloutsos, Mudumbai Ranganathan, and Yannis Manolopoulos. 1994. Fast subsequence matching in time-series databases. *Acm Sigmod Record* 23, 2 (1994), 419–429.

[16] Myron Flickner, Harpreet Sawhney, Wayne Niblack, Jonathan Ashley, Qian Huang, Byron Dom, Monika Gorkani, Jim Hafner, Denis Lee, Dragutin Petkovic, et al. 1995. Query by image and video content: The QBIC system. *computer* 28, 9 (1995), 23–32.

[17] Daniel Y Fu, Will Crichton, James Hong, Xinwei Yao, Haotian Zhang, Anh Truong, Avanika Narayan, Maneesh Agrawala, Christopher Ré, and Kayvon Fatahalian. 2019. Rekall: Specifying video events using compositions of spatiotemporal labels. *arXiv preprint arXiv:1910.02993* (2019).

[18] Hector Garcia-Molina. 2008. *Database systems: the complete book*. Pearson Education India.

[19] Teofilo F Gonzalez. 1985. Clustering to minimize the maximum intercluster distance. *Theoretical computer science* 38 (1985), 293–306.

[20] Antonin Guttman. 1984. R-trees: A dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD international conference on Management of data*. 47–57.

[21] John Michael Hammersley and David Christopher Handscomb. 1964. General principles of the Monte Carlo method. In *Monte Carlo Methods*. Springer, 50–75.

[22] Sariel Har-Peled and Soham Mazumdar. 2004. On coresets for k-means and k-median clustering. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*. 291–300.

[23] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2017. Mask r-cnn. In *ICCV*. IEEE, 2980–2988.

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*. 770–778.

[25] Joseph M Hellerstein, Jeffrey F Naughton, and Avi Pfeffer. 1995. *Generalized search trees for database systems*. September.

[26] Kevin Hsieh, Ganesh Ananthanarayanan, Peter Bodik, Paramvir Bahl, Matthai Philipose, Phillip B Gibbons, and Onur Mutlu. 2018. Focus: Querying Large Video Datasets with Low Latency and Low Cost. *OSDI* (2018).

[27] Stratos Idreos, Martin L Kersten, Stefan Manegold, et al. 2007. Database Cracking.. In *CIDR*, Vol. 7. 68–78.

[28] Hosagrahar V Jagadish, Beng Chin Ooi, Kian-Lee Tan, Cui Yu, and Rui Zhang. 2005. iDistance: An adaptive B+-tree based indexing method for nearest neighbor search. *ACM Transactions on Database Systems (TODS)* 30, 2 (2005), 364–397.

[29] Junchen Jiang, Ganesh Ananthanarayanan, Peter Bodik, Siddhartha Sen, and Ion Stoica. 2018. Chameleon: scalable adaptation of video analytics. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*. ACM, 253–266.

[30] Daniel Justus, John Brennan, Stephen Bonner, and Andrew Stephen McGough. 2018. Predicting the computational cost of deep learning models. In *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 3873–3882.

[31] Daniel Kang, Peter Bailis, and Matei Zaharia. 2019. BlazeIt: Optimizing Declarative Aggregation and Limit Queries for Neural Network-Based Video Analytics. *PVLDB* (2019).

[32] Daniel Kang, John Emmons, Firas Abuzaid, Peter Bailis, and Matei Zaharia. 2017. NoScope: optimizing neural network queries over video at scale. *PVLDB* 10, 11 (2017), 1586–1597.

[33] Daniel Kang, Edward Gan, Peter Bailis, Tatsunori Hashimoto, and Matei Zaharia. 2020. Approximate Selection with Guarantees using Proxies. *PVLDB* (2020).

[34] Daniel Kang, John Guibas, Peter Bailis, Tatsunori Hashimoto, Yi Sun, and Matei Zaharia. 2021. Accelerating Approximate Aggregation Queries with Expensive Predicates. *PVLDB* (2021).

[35] Daniel Kang, Ankit Mathur, Teja Veeramacheneni, Peter Bailis, and Matei Zaharia. 2021. Jointly Optimizing Preprocessing and Inference for DNN-based Visual Analytics. *PVLDB* (2021).

[36] Qiuqiang Kong, Yin Cao, Turab Iqbal, Yuxuan Wang, Wenwu Wang, and Mark D Plumbley. 2020. Panns: Large-scale pretrained audio neural networks for audio pattern recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 28 (2020), 2880–2894.

[37] Kevin Lin, Huei-Fang Yang, Jen-Hao Hsiao, and Chu-Song Chen. 2015. Deep learning of binary hash codes for fast image retrieval. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 27–35.

[38] Yao Lu, Aakanksha Chowdhery, Srikanth Kandula, and Surajit Chaudhuri. 2018. Accelerating Machine Learning Inference with Probabilistic Predicates. In *SIGMOD*. ACM, 1493–1508.

[39] Ozan Sener and Silvio Savarese. 2017. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489* (2017).

[40] John R Smith and Shih-Fu Chang. 1997. VisualSEEk: a fully automated content-based image query system. In *Proceedings of the fourth ACM international conference on Multimedia*. 87–98.

[41] Jeffrey D Ullman. 1984. *Principles of database systems*. Galgotia publications.

[42] Kilian Q Weinberger and Lawrence K Saul. 2009. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research* 10, 2 (2009).

[43] Tiantu Xu, Luis Materon Botelho, and Felix Xiaozhu Lin. 2019. VStore: A Data Store for Analytics on Large Videos. In *Proceedings of the Fourteenth EuroSys Conference 2019*. ACM, 16.

[44] Cui Yu, Beng Chin Ooi, Kian-Lee Tan, and HV Jagadish. 2001. Indexing the distance: An efficient method to knn processing. In *VLDB*, Vol. 1. 421–430.

[45] Yuhao Zhang and Arun Kumar. 2019. Panorama: a data system for unbounded vocabulary querying over video. *Proceedings of the VLDB Endowment* 13, 4 (2019), 477–491.

[46] Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning. *CoRR* abs/1709.00103 (2017).