Task-agnostic Indexes for Deep Learning-based Queries over Unstructured Data

Daniel Kang*, John Guibas*, Peter Bailis, Tatsunori Hashimoto, Matei Zaharia
Stanford University

ABSTRACT

Unstructured data is now commonly queried by using *target* deep neural networks (DNNs) to produce structured information, e.g., object types and positions in video. As these target DNNs can be computationally expensive, recent work uses proxy models to produce *query-specific proxy scores*. These proxy scores are then used in downstream query processing algorithms for improved query execution speeds. Unfortunately, proxy models are often trained *per-query*, require large amounts of training data from the target DNN, and new training methods per query type.

In this work, we develop an index construction method (task-agnostic semantic trainable index, TASTI) that produces reusable embeddings that can be used to generate proxy scores for a wide range of queries, removing the need for query-specific proxies. We observe that many queries over the same dataset only require access to the schema induced by the target DNN. For example, an aggregation query counting the number of cars and a selection query selecting frames of cars require only the object types per frame of video. To leverage this opportunity, TASTI produces embeddings per record that have the key property that close embeddings have similar extracted attributes under the induced schema. Given this property, we show that clustering by embeddings can be used to answer downstream queries efficiently. We theoretically analyze TASTI and show that low training error guarantees downstream query accuracy for a natural class of queries. We evaluate TASTI on four video and text datasets, and three query types. We show that TASTI can be 10× less expensive to construct than proxy models and can outperform them by up to 24× at query time.

PVLDB Reference Format:
Daniel Kang*, John Guibas*, Peter Bailis, Tatsunori Hashimoto, Matei Zaharia . Task-agnostic Indexes for Deep Learning-based Queries over Unstructured Data. PVLDB, 14(1): XXX-XXX, 2020.
doi:XX.XX/XXX.XX

1 INTRODUCTION

Unstructured data such as video and text are becoming increasingly feasible to analyze due to automatic methods of analysis in the form of deep neural networks (DNNs). A common approach of analysis is to use a *target DNN* to extract structured information into an *induced schema* from this data. For example, object detection DNNs [34] can be used to extract the list of object types and positions of a frame of video. This information can then be used to answer a range of queries, such as counting the number of cars per frame. However, state-of-the-art target DNNs can be prohibitively expensive to execute on large data volumes, executing as slow as 3 frames per second (fps), or 10× slower than real time [21].

To reduce the cost of analysis over unstructured data, recent work has proposed *query-specific proxy models* to approximate target DNNs. For example, low cost proxy models can be used for selecting unstructured data records matching predicates [3, 24, 30, 31, 33], for aggregation queries [29], and for limit queries [29]. For each query, a new proxy model is used to generate *proxy scores* per data record, in which the goal is to approximate the result of executing the target DNN the data record for the particular query; these scores are subsequently used for query processing (e.g., BlazeIt debiases proxy scores with the method of control variates [19, 29]).

Unfortunately, each query type often requires large amounts of training data from the target DNN and new, ad-hoc training procedures [3, 29–31, 33]. In particular, methods based on query-specific proxy models have three key drawbacks. First, obtaining large amounts of training data from the target DNN can be computationally expensive. For example, BlazeIt and NoScope require up to 150,000 annotations from the target DNN [29, 30] and other systems require expensive human annotations [3, 24, 33]. 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 queries. Thus, implementing queries can be challenging and computationally expensive at ingest time (i.e., obtaining target DNN annotations).

This prior work ignores a key opportunity for accelerating queries: the target DNN induces a schema that can be shared across many query types. For example, an aggregation query and selection query over cars in a video only require information of object types and positions. This schema is thus a sufficient statistic that can be used to answer many query types. While fully materializing this sufficient statistic is expensive, query processing systems would ideally leverage this sufficient statistic to avoid repeated work and target DNN invocations.

To address these issues and leverage this opportunity, we propose **T**ask-**A**gnostic **S**emantic **T**rainable **I**ndexes (TASTI), a method of indexing unstructured data via embeddings for accelerating downstream proxy score-based queries over the induced schema. Given the target DNN and a user-provided notion of "closeness" over the schema, TASTI produces semantic embeddings for each unstructured data record (e.g., frame of video or line of text), with the desideratum that records with close embeddings are similar under the induced schema for all downstream queries. TASTI requires that the induced schema has a notion of distance, e.g., that frames of a video with similar object types and object positions are close. Given these embeddings and a small set of records annotated by the target DNN, we demonstrate how to answer queries over the schema efficiently. We show that TASTI can be simultaneously

* Marked authors contributed equally.

Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

up to 10× more sample efficient to construct and up to 24× more efficient at query time than recent state-of-the-art.

To generate these indexes in a sample-efficient manner, TASTI uses a triplet loss training procedure to train an embedding DNN. TASTI then uses this DNN to produce embeddings per data record and annotates a small set of records with the target DNN ("cluster representatives"), which are stored as the index. The embedding DNN is task-agnostic, where "task" refers to a query over the induced schema and "agnostic" is in the sense of being agnostic to any downstream queries over the induced schema. Our sample-efficient usage of the triplet loss [38] can use as much as 10× fewer target-DNN annotations to achieve similar quality results to proxy model-based methods [29–31]. Furthermore, if training is expensive, we show that pre-trained DNNs can be used to produce embeddings that weakly satisfy our desideratum.

TASTI can be used in conjunction with any query processing algorithm that requires access to proxy scores. We demonstrate how to answer aggregation, selection, and limit queries based on proxy algorithms [3, 29–31, 33] in this work. To answer queries given the cluster representatives and embeddings in TASTI, we cluster the remaining frames by embedding distance and propagate the annotations to unannotated records (Section 4). For example, for counting the number of cars in a video, we would assign an unannotated frame the average number of cars of the closest cluster representatives. These scores are then used in query processing algorithms, often to debias proxy scores. We further show that as the target DNN is executed over the data, we can further cluster the embeddings, which improves performance (i.e., TASTI's indexes can be "cracked" [25]).

To understand TASTI's performance, we provide a theoretical analysis of how the triplet loss corresponds to downstream query accuracy (Section 5). We prove a positive result: queries that compute Lipschitz-continuous functions of the data will achieve *exact* results for 0 triplet loss and dense enough clustering. We further prove bounds on query performance when the triplet loss is not 0.

We implement TASTI in a prototype system. To demonstrate the efficacy of our indexing method, we evaluate TASTI on four datasets, including widely studied video datasets [9, 27, 29, 31, 39] and a text dataset [42]. We execute aggregation, selection, and limit queries over these datasets, as studied by prior work [29–31]. Across all queries and datasets, we show that TASTI outperforms state-of-the-art [29, 31] by up to 24×. We further show that TASTI performs well across a range of parameter settings.

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 why our algorithms outperform baselines.
(3) We evaluate TASTI on four unstructured datasets and three query types, showing it can outperform state-of-the-art.

2 OVERVIEW AND EXAMPLE

2.1 Overview

TASTI is primarily designed for batch analytics over unstructured data. As its primary input, TASTI takes a target DNN that extracts structured information from unstructured data records, as many contemporary systems do [3, 24, 29–31, 33]. Given the schema induced by the target DNN, TASTI will build an index consisting of per-record embeddings that places input records with similar extracted structure together and a set of cluster representatives (sample records annotated by the target DNN). This index can then be used to accelerate queries over the induced schema.

As an example, consider queries over object types and object positions in video. An object detection DNN [21] could be used as a target DNN to extract this information. Given the target DNN, TASTI will construct an index that places frames with similar object types and object positions together. Users can then issue queries selecting events of cars or counting the number of cars. As another example, consider a dataset that contains natural language questions. A semantic parsing target DNN could extract SQL statements that answer the question; TASTI can construct an index that places questions with similar SQL operators and similar predicates together. Users can then issues queries to understand natural language questions, e.g., to count the average number of predicates.

To accelerate queries over the induced schema, TASTI propagates the cluster representatives' annotations via embedding distance to the remainder of the records, producing approximate scores. These approximate scores are used in query processing, e.g., in algorithms to efficiently execute aggregation and selection queries.

We now describe the specific problem statement, index construction, and query processing procedures.

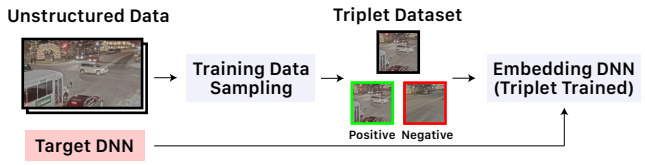
Problem statement. As input, TASTI takes a target DNN, an induced schema over the target DNN outputs, a target DNN invocation budget for index construction, a user-provided k closest distances to store, and a notion of closeness over the induced schema. TASTI's indexes have the desideratum that "close" records are close in embedding space and vice versa for records that are far. The primary cost in index construction are the target DNN invocations used for training and annotating cluster representatives; TASTI will attempt to construct high quality indexes subject to the budget.

TASTI's primary goal is to produce high quality proxy scores for query processing algorithms that require such scores. High quality typically refers to high correlation between proxy scores and target DNN outputs. We demonstrate how to generate such proxy scores for selection queries [3, 30, 31, 33], aggregation queries [29], and limit queries [29].

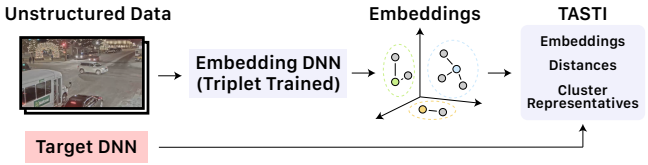
Index construction. TASTI's indexes consist of semantic embeddings per data record, a small set of annotated *cluster representatives*, and distances from data records to cluster representatives.

To generate the embeddings, TASTI can train a DNN in a task-agnostic manner, such that semantically similar data records have similar embeddings, as measured by ℓ_2 distance (Figure 1a). TASTI can also use embeddings from a pre-trained DNN to reduce index construction time (although this may degrade query-time performance). For the video example, TASTI's embedding DNN will attempt to cluster frames with similar object types and object positions together. Perhaps surprisingly, TASTI can also perform well using pre-trained DNNs with no additional training; we present results using pre-trained DNNs as "TASTI-PT" in Section 6.

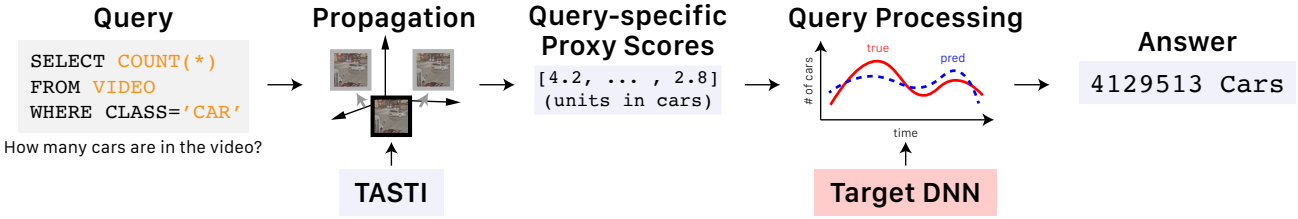
The cluster representatives can be selected in any way, but in this work, we use the furthest point first (FPF) algorithm [17]. We use FPF as it performs well in practice and because our theoretical



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



(b) Overview of TASTI's index construction procedure. TASTI produces embeddings per data record, selects sample records to annotate, and computes embedding distances to those records.



(c) Overview of TASTI's query processing. Given a query, TASTI will propagate query-specific scores to unannotated records. These scores are coalesced into a query-specific proxy scores, which are used in downstream query processing algorithms.

Figure 1: TASTI system overview

analysis relies on the maximum intra-cluster distance, which the FPF provides a guarantee on. TASTI subsequently computes the distances of all embedded records to the cluster representatives and stores the rank ordering by distance (Figure 1b). TASTI can also improve its index as the target DNN is invoked over the data as queries are issued (i.e., TASTI's indexes can naturally be cracked [25]). We describe full details in Section 4.

Query processing. In this work, we show how to use TASTI's indexes to generate proxy scores for query processing algorithms that require such scores, specifically for aggregation [29], selection [3, 30, 31, 33], and limit queries [29].

We show an overview of TASTI's query processing procedure in Figure 1c. Given the structured output of the target DNN on the cluster representatives, TASTI will produce proxy scores on these records and propagate scores to the remainder of the records. For example, suppose the user issued an aggregation query for the number of cars in a frame. On the cluster representatives, TASTI will return the number of cars as predicted by the target DNN as the score and these scores will be propagated to the remainder of the frames. Downstream query processing occurs over the scores; some query processing algorithms will additionally query the target DNN. We describe full details in Section 4.

In contrast, prior work uses query-specific ad-hoc proxy models to generate proxy scores. We show that TASTI can generate higher quality proxy scores, which results in more accurate or faster downstream query processing (Section 6).

2.2 Example

As a concrete example, consider constructing an index for visual data, in which queries over object types and positions are issued. In this case, the target DNN 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) and one that selects frames with cars (selection query). To understand how the index construction procedure and query processing works, we describe the intuition below.

Index construction. Each target DNN and induced schema requires a heuristic of "close" and "far" records, either as a Boolean function or as a cutoff based on a continuous distance measure. One such heuristic for this 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 IsClose(frame1: List[Boxes], frame2: List[Boxes])
  -> bool:
    if len(frame1) != len(frame2):
        return False
    return all_close(frame1, frame2)
```

where `all_close` is a helper function that returns true if all boxes in `frame1` have a corresponding "close" box in `frame2`. Given the notion of close frames, TASTI will train an embedding DNN via the triplet loss, which separates "close" and "far" frames. Then, TASTI will compute embeddings over all frames of the video and select a set of frames to annotate with the target DNN. In this case, TASTI will store the object types and positions for the cluster representatives as predicted by the target DNN.

Given the annotated frames, TASTI will compute the embedding ℓ_2 distance from all other frames to these annotated frames. TASTI will also store the k closest frames. These distances will be subsequently used in query processing.

Query processing. Given the structured records from the target DNN output, TASTI will generate scores on the cluster representatives. Then, TASTI will propagate these scores to the remainder of the records. While TASTI provides a default method for doing so, a developer can also provide custom functions for propagation. We describe the intuition behind two example queries.

Approximate aggregation. Suppose the user issues a query for the average number of cars per frame, as studied by BlazeIt [29]. To optimize this query, BlazeIt trains a proxy model to estimate the

number of cars per frame. This proxy model is then used to generate a query-specific proxy score per frame, which is the estimate from the proxy model. BlazeIt then uses these scores as a "control variate" [19] to reduce the variance in sampling.

In contrast, TASTI computes the query-specific proxy score as the weighted average of the number of cars in the k closest annotated frames (we defer pseudocode to Section 4). This will produce an estimate of the number of cars in a given unannotated frame. These scores are then coalesced and can be used by BlazeIt's query processing algorithm to optimize approximate aggregation.

Approximate selection. Suppose the user issues a query to select 90% of frames with cars with 95% probability of success, as studied by "recall target" setting in SUPG [31]. SUPG requires a probability that a frame contains a car. As with BlazeIt, SUPG will use a proxy model to estimate whether a record matches the predicate.

To compute this probability (i.e., query-specific proxy score), 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.

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 task-agnostic indexes given the target DNN. Recall that 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 notion of closeness. 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 and the index construction in Figure 1b. We note that TASTI's training procedure is optional: pre-trained embeddings can be also be used for the index if training is expensive.

3.1 Training

TASTI optionally trains a task-agnostic mapping between data records (e.g., frames of a video) and semantic, task-agnostic 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, 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.

Domain-specific triplet loss. To train the embedding DNN to produce embeddings that fulfill our desiderata, TASTI uses the triplet loss [38]. 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. Formally, the per-example triplet loss is defined as

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

for some embedding function ϕ.

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. 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 DNN to determine whether pairs of records are close or not. Due to the cost of the target DNN, TASTI must sample records to be selected for training; we assume the user provides a budget of target DNN 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).

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 DNN pre-trained on other semantic data to select such diverse points. We note that these pre-trained DNNs are widely available, e.g., DNNs pre-trained on ImageNet [22] or on large text corpora (BERT) [12]. Pre-trained DNNs produce embeddings that are typically semantically meaningful, although typically not adapted to the specific set of queries.

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 DNN annotations.

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 deciding 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 for cluster representatives may do well on average at query time, it may perform poorly on rare events.

To address this issue, TASTI selects cluster representatives via furthest-point first (FPF). FPF iteratively chooses the furthest point from the existing set of cluster representatives as the newest representative. FPF is both computationally efficient and provides a 2-approximation to the optimal maximum intra-cluster distance. Intuitively, FPF chooses points that are diverse in embedding space. 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

In contrast to prior work, which can only share work between queries in an ad-hoc manner, TASTI can naturally be extended with "cracking" functionality [25]. In particular, when any query executes the target DNN, TASTI can cache the target DNN result. The records over which the target DNN are executed over can then be added as new cluster representatives. Computing the distance to the new cluster representative is computationally efficient and trivially parallelizable.

3.4 Performance Analysis

Suppose there are N data records, D dimensions, L training iterations, and a total target DNN budget of C. Denote the costs of the target DNN, embedding DNN, and distance computation as c_T, c_E, and c_D respectively. Then, 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 [28].

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 DNN. For example, the Mask R-CNN we use in this work executes at 3 fps, compared to the embedding DNN which executes at 12,000 fps.

4 QUERY PROCESSING

Given an index, how can TASTI accelerate query processing? To execute a query, TASTI will construct *query-specific proxy scores* of the data records that 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 DNN on the data records for the particular query. Consider an aggregation query counting the average number of cars per frame [29]. In this case, the query-specific proxy scores would be an estimate of the number of cars in a given frame.

Many downstream query processing techniques only require query-specific proxy scores and the target DNN. For example, certain types of selection without guarantees (e.g., binary detection) [3, 30, 33], selection with statistical guarantees [31], aggregation [29], and limit queries [29] only require query-specific proxy scores and the target DNN. We describe concrete examples in Section 4.3.

We assume TASTI is provided methods that take target DNN outputs and produce a numeric score; this can be done automatically in many cases, but a developer may also implement custom functions produce such scores. We describe the interface for these scores, describe how TASTI propagates query-specific proxy scores, and give examples of uses of proxy scores for query processing. We show a schematic of the query processing procedure in Figure 1c.

4.1 Query-specific Score

In order to execute queries, a developer must specify a query-specific scoring function, which takes the output of the target DNN and returns a query-specific proxy score. We note that query processing systems must implement such a function even without using TASTI's indexes, so this is a natural requirement. Concretely, consider the query of counting the average number of cars per frame. The scoring function would return the number of cars as predicted by the target DNN.

The API for specifying scoring functions is as follows. Denote the type of the output of the target DNN as `TargetDNNOutput` (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: TargetDNNOutput) -> ScoreType
```

These functions can often be implemented in few lines of code. Concretely, 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.

4.2 Score Propagation

Given the query-specific scoring functions, TASTI will execute the scoring functions on the cluster representatives (as the target DNN 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. Given k, the score for each data record will be the 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.

4.3 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 [29]. 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, 30, 31, 33]. 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 to used achieve statistical guarantees on the recall or precision of selected records [31]. These scores can also be used to 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, 30, 33].

Limit queries. Consider a query that selects 10 frames containing at least 5 cars [29]. 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 DNN as ordered by the query-specific proxy scores. The algorithm will terminate once the requested number of frames is found.

4.4 Performance Analysis

We use the notation of Section 3.4. Computing the query-specific proxy scores requires C calls of the scoring function and $O(N \cdot k)$ arithmetic operations, as distances to cluster representatives are cached. In all applications we consider, this procedure is orders of magnitude more efficient than executing the target DNN, which often requires billions of floating point operations on an accelerator.

5 THEORETICAL ANALYSIS

We present a statistical performance analysis of our index construction and query processing 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. We formalize this intuition by analyzing how downstream queries behave under the triplet loss. We specifically analyze the case where $k = 1$.

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

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

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. As a concrete example, consider the video example described in Section 2. $\mathcal{D}$ is the set of frames, ϕ is the trained embedding DNN, and we use the metric induced by the notion of closeness 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 Understanding the 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.

Theorem and proof. We now prove 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*

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

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:

$$|\ell_Q(x, f(x)) - \ell_Q(x, \hat{f}(x))| \tag{3}$$
$$\leq |\ell_Q(x, f(x)) - \ell_Q(c(x), f(c(x)))| + |\ell_Q(c(x), f(c(x))) - \ell_Q(x, f(c(x)))| \tag{4}$$
$$\leq M \cdot K_Q \tag{5}$$

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. □

5.3 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*

$$\mathbb{E}[\ell_Q(x, \hat{f}(x))] \leq \mathbb{E}[\ell_Q(x, f(x))] + M \cdot K_Q + \frac{C \sup_x |\bar{B}_M(x)|}{m} \alpha. \tag{6}$$

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

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

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.

□

5.4 Discussion

We have shown that many classes of queries will have bounded loss (i.e., discrepancy from using exact answers). However, we note that our analysis has several gaps from our exact procedure. First, we use the nearest k cluster representatives to generate the query-specific proxy scores. 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 evaluate TASTI on four real world video and text datasets. We describe the experimental setup and baselines. We then demonstrate that TASTI's index construction is less expensive than recent state-of-the-art, that it outperforms on all settings we consider, that all components of TASTI are required for performance, and that TASTI is not sensitive to hyperparameter settings.

6.1 Experimental Setup

Datasets, target DNNs, and triplet loss. We consider three video datasets and one text dataset. We use the `night-street`, `taipei`, and `amsterdam` videos as used by BlazeIt [29]. `night-street` is widely used in video analytics evaluations [9, 29, 30, 39] and `taipei` has two object classes (car and bus). We use Mask R-CNN as the target DNN. We use ResNet-18 as our embedding DNN. The triplet loss separates frames with objects that are far apart and frames with different numbers of objects.

For the text dataset, we use a semantic parsing dataset [42]. The dataset consists of pairs of natural language questions and corresponding SQL statements. We assume the SQL statements are not known at query time and must be annotated by crowd workers (i.e., that crowd workers are the "target DNN"). We use BERT [12] for the embedding DNN. We consider queries over SQL operators and number of predicates. The triplet loss separates questions over different operators and number of predicates.

Queries and metrics. We evaluate TASTI and recent state-of-the-art on three general classes of queries: aggregation, selection, and limit queries.

For aggregation queries, we query for the approximate count of some statistic of the target DNN executed on the unstructured data records. We compute the average number of objects per frame for the video datasets and the average number of predicates per query for the WikiSQL dataset. For all methods and datasets, we use the EBS sampling as used by the BlazeIt system [29], which provides guarantees on error. EBS sampling is adaptive with respect to how well the query-specific proxy scores are correlated with the target DNN, so better proxy scores will result in fewer target DNN invocations. As a result, we measure the number of target DNN invocations (lower is better).

For selection queries, we execute approximate selection queries with recall targets (SUPG queries [31]). Given a target DNN invocation 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 failure"): these queries are useful in scientific applications or mission-critical settings [31]. In contrast to queries that do not provide statistical guarantees, SUPG guarantees the recall target is satisfied with high probability. Furthermore, in contrast to limit queries, SUPG guarantees the recall of the returned set, as opposed to returning a set number of records. We select for frames with objects for video datasets and natural language questions that are parsed into selection SQL statements for the WikiSQL dataset. Since SUPG queries fix the number of target DNN invocations, we measure the false positive for the recall target settings, respectively (lower is better).

For limit queries, we use the ranking algorithm proposed in [29]. This ranking algorithm will examine 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. This algorithm will execute the target DNN on all proposed frames until the requested number of records are found. As such, we measure the number of target DNN invocations (lower is better).

Methods evaluated. We use the exact setup of BlazeIt and SUPG for the respective query types where possible. These systems train a proxy model at query time using an ad-hoc loss. We use the exact proxy models for the video datasets (a "tiny ResNet") and FastText embeddings [7] for the WikiSQL 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 evaluate 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 time the end-to-end time of process of loading video and executing the embedding DNN for measuring wall clock times for TASTI.

Due to the large cost of executing the target DNN, we cache target DNN results and compute the average execution time for the target DNN. For baselines, we only time the target DNN computation and exclude the computational cost of proxy models, which strictly improves the baselines. We exclude the cost of query processing as it is negligible in all cases (over 3 orders of magnitude less expensive for all queries we consider).

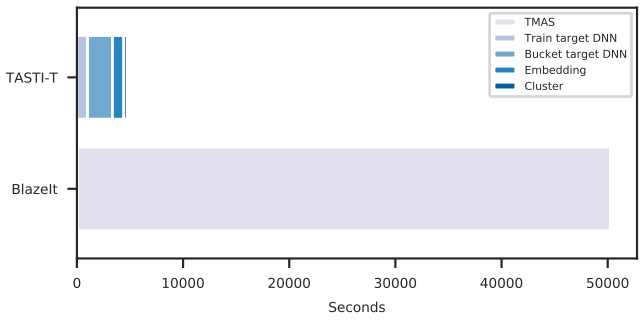


Figure 2: Breakdown of time to construct indexes for TASTI and for BlazeIt on the `night-street` dataset. The BlazeIt index is the "target-model annotated set" (TMAS) [29]. Similar results hold for other datasets.

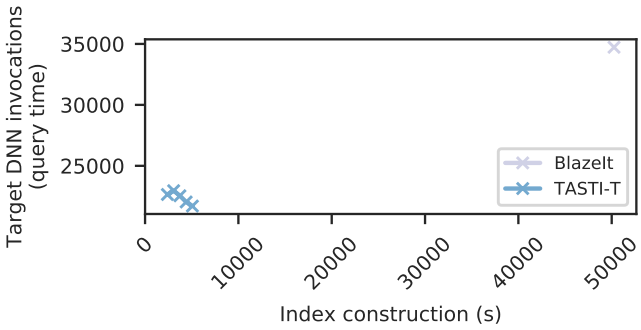


Figure 3: Index construction time vs performance of TASTI and BlazeIt for aggregation queries on the `night-street` dataset. Similar results hold for other datasets.

6.2 Index Construction Performance

To understand the index construction performance, we measure the wall clock time to construct TASTI indexes. We compare to BlazeIt, which effectively constructs indexes by executing the target DNN on a subset of the data (referred to as the "TMAS" in [29]). For BlazeIt, we only consider the cost of constructing the TMAS. For TASTI, we measure the full index construction time, including the embedding DNN time and the distance computation time. We compute 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. As shown, TASTI is substantially faster than BlazeIt, costing 10× less. The reduced cost is due to fewer invocations from the target DNN. Similar results hold for other datasets. We additionally show the index construction time vs performance for BlazeIt and a range of parameters for TASTI. We show results in Figure 3. As shown, TASTI can outperform or match BlazeIt performance with up to 10× less expensive index construction times.

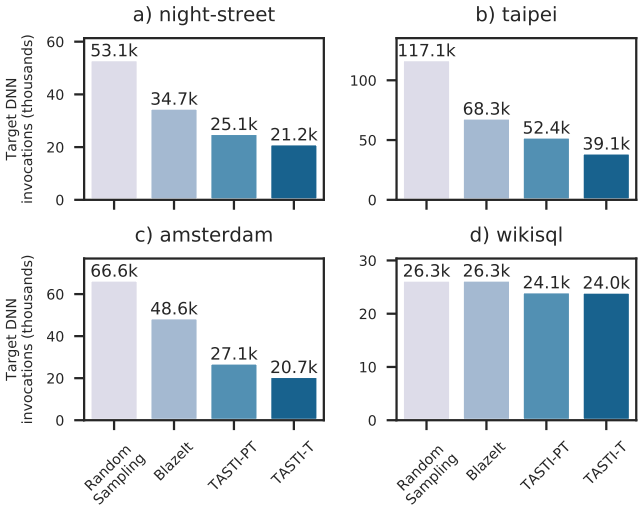


Figure 4: Number of target DNN invocations for baselines and TASTI for aggregation queries (lower is better). As shown, TASTI outperforms baselines in all cases, including prior state-of-the-art by up to 2×.

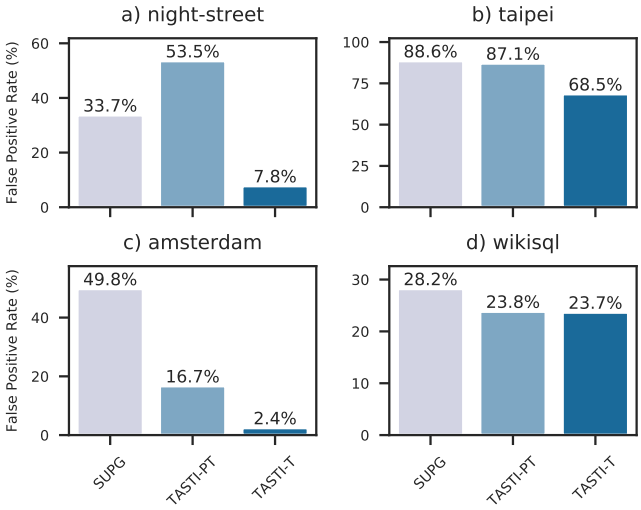


Figure 5: False positive rate for recall-target SUPG queries (lower is better). We show the performance of baselines and TASTI. As shown, TASTI outperforms baselines in all cases.

6.3 End-to-end Performance

We show that TASTI outperforms recent state-of-the-art for approximate aggregation, selection with guarantees, and limit queries. For all video datasets in this Section, we use 3,000 training records, 7,000 cluster representatives, and an embedding size of 128. For the WikiSQL dataset, we use 500 training examples and 500 cluster representatives. For the remainder of the experiments, we do not include the cost of constructing the index. Other work also excludes this cost [3, 29–31, 33], and TASTI can construct indexes substantially faster than prior work.

Approximate aggregation. For approximate aggregation queries, we compare TASTI (with and without training) to using standard random sampling and an ad-hoc trained proxy model. We use the exact experimental setup as BlazeIt [29] for video datasets (the exact code) and BERT embeddings for the WikiSQL dataset. We aggregate over the average number of cars per frame for all video datasets (TASTI also outperforms on bus for `taipei`).

We show results in Figure 4. As shown, TASTI outperforms for aggregation queries on all datasets. In particular, TASTI outperform recent state-of-the-art optimized for aggregation queries (BlazeIt) by up to 2× while maintaining less expensive index construction costs. Further, compared to standard random sampling, TASTI outperforms 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 DNN 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 compare TASTI (with and without training) to using an ad-hoc trained proxy model (standard random sampling is not appropriate for SUPG queries). We use the exact same experimental setup as in SUPG [31] for the video datasets and BERT embeddings for the WikiSQL dataset. For all queries, we use a recall target of 90% with a confidence of 95%, as used in [31]. We search for cars in `night-street` and `amsterdam`, buses in `taipei` (TASTI also outperforms for cars), and star operators for WikiSQL.

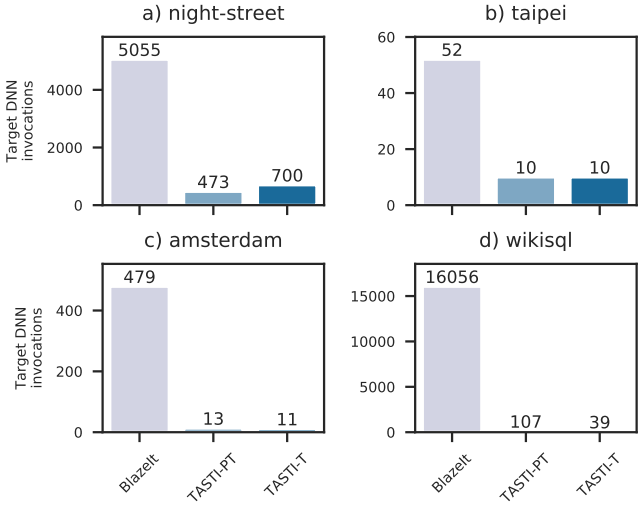


Figure 6: Number of target DNN invocations for baselines and TASTI for limit queries (lower is better). As shown, TASTI outperforms baselines in all cases, including prior state-of-the-art by up to 34×.

As shown in Figure 5, TASTI outperforms on all datasets. In particular, TASTI can improve the false positive rate by almost 2× over recent state-of-the-art. We further show that the triplet training 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).

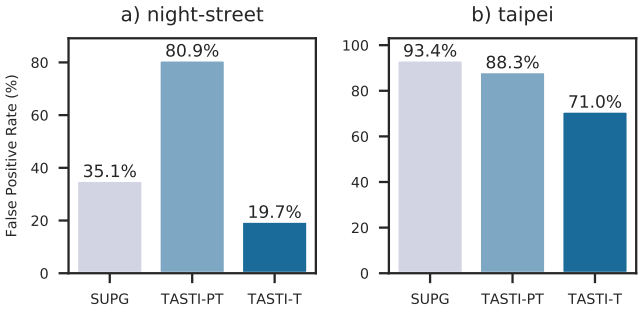


Figure 7: SUPG queries for selecting objects of interest on the left hand side of the frame. This query violates the Lipschitz condition, but TASTI still outperforms baselines.

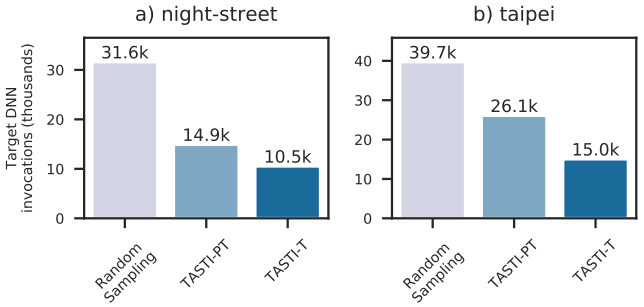


Figure 8: Aggregation query for the average x position of objects in frames of a video. Recent state-of-the-art is not well suited for this query as regression can be difficult for proxy models. In contrast, TASTI performs well on these queries.

Limit queries. For limit queries, we use the ranking algorithm proposed by BlazeIt [29]. We use the exact same experimental setup as BlazeIt [29] for the video datasets (including the query configurations, e.g., number of objects, etc.) and BERT embeddings for the WikiSQL dataset. 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.

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

6.4 New Queries

In addition to the queries above, we demonstrate that TASTI can be used to efficiently answer queries that prior work is not well suited for. We consider 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 consider 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 compare TASTI to to training a proxy model by extending SUPG and to TASTI without triplet training. Results are shown in Figure 7.

We note that 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 this query as it uses the information from the target DNN, 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 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. We note that 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 1: 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 1. As shown, TASTI outperforms on quality metrics for all settings we considered, indicating that TASTI's proxy scores are higher quality.

6.6 Cracking

We further demonstrate that TASTI's index can easily be used for "cracking" [25]. To show this, we executed an aggregation query followed by a SUPG query and vice versa. We used the target DNN 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.

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 2: Performance of TASTI after cracking. We measured query performance of a SUPG/aggregation query after cracking (false positive rate and number of target DNN 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.

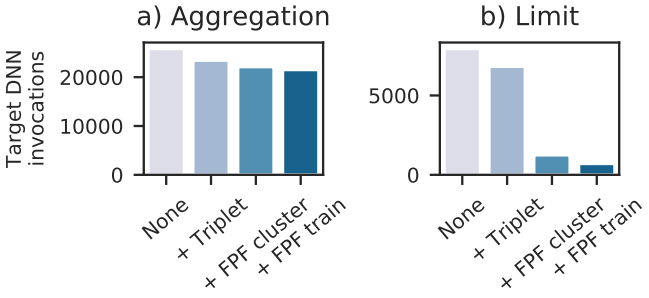


Figure 9: Factor analysis, in which optimizations are added in in sequence. As shown, all optimizations improve performance for aggregation queries. For limit queries, FPF training and clustering are required for triplet training to improve performance.

As shown in Table 2, 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 `night-street` for aggregation and limit queries. Aggregation queries highlight "average-case" performance and limit queries highlight "rare-event" performance. We choose `night-street` as it has been widely studied in visual analytics [9, 29–31, 39]; 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.

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 `night-street` for aggregation and limit queries; other datasets have similar behaviors.

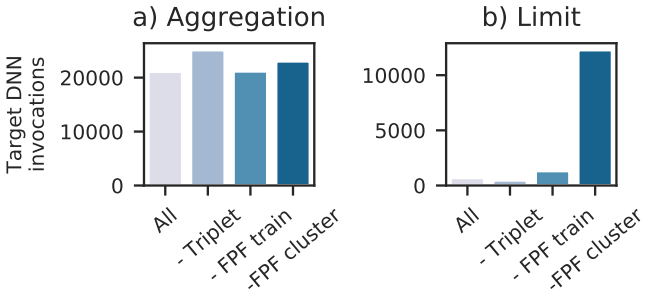


Figure 10: Lesion study, in which optimizations are removed individually (they are not removed cumulatively). As shown, all optimizations improve performance. Recall that lower is better for both aggregation and limit queries.

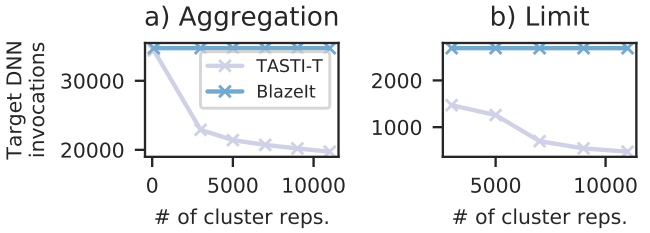


Figure 11: Number of cluster representatives vs performance on aggregation and limit queries on the `night-street` dataset. As shown, TASTI outperforms baselines on a range of parameter settings.

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 use 3,000, 5,000, 7,000, 9,000, and 11,000 buckets for both queries. For aggregation queries, we additionally use 50 buckets.

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`

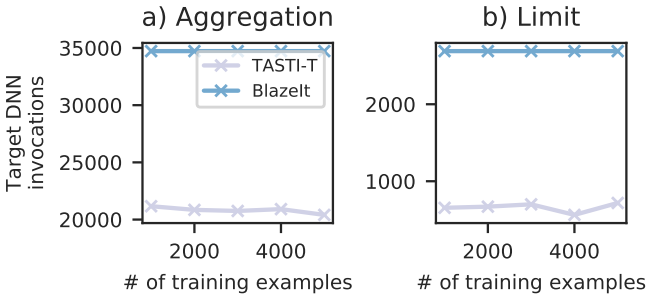


Figure 12: Number of training examples vs performance on aggregation and limit queries on the `night-street` dataset. As shown, TASTI outperforms baselines on a range of parameter settings.

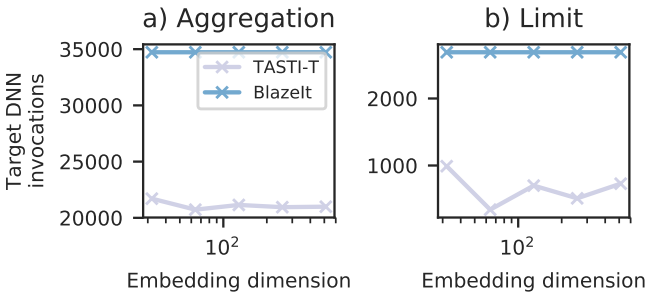


Figure 13: Embedding size vs performance on aggregation and limit queries on the `night-street` dataset. TASTI outperforms baselines on a range of parameter settings.

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 TASTI's 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, aggregation performance does not substantially change with the embedding size. However, the performance of SUPG precision target queries does vary with the embedding size. We hypothesize the difference primarily comes from downstream query processing procedures. For aggregation queries, only the correlation of the proxy and target DNN affects performance. In contrast, the calibration affects the performance of SUPG queries. As the embedding size decreases, we hypothesize that the aggregate correlation remains approximately constant, but the calibration may change. We defer further investigation to future work.

7 RELATED WORK

Structured indexes. There is a long history in the database literature of indexes for structured data [37]. These indexes generally are used to accelerated lookups on certain columns. Techniques range from tree-like structures [6, 11, 23] to hash tables [16]. However, these indexes assume that the data is present in a structured format, which is not the case for the data we consider.

Unstructured 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 [10, 18], time series [2, 13], and low-level visual features [14, 36]. Other indexes accelerate KNN search in possibly high dimensions, when the distances are meaningful [26, 40]. Work in retrieval has used indexed embeddings to accelerate search for semantically similar items, in particular for visual data [4, 32, 41]. 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 DNNs.

Coresets. Several communities, including the theory and deep learning communities, have considered coresets [1]. Coresets are concise summaries of data. They have been used for nearest neighbor searches [20], streaming data [8], active learning [35], and other applications. We are unaware of work that trains embedding DNNs in a task-agnostic manner for the construction of indexes.

DNN-based queries. Recent work in the database 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, 24, 30, 33], selection with statistical guarantees [31], aggregation queries [29], limit queries [29], tracking queries [5], and a range of other queries. These systems aim to reduce the cost of expensive target DNNs, 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 DNN. We leverage many of the downstream query processing techniques used in this prior work.

Other work assumes that the target DNN is not expensive to execute or that extracting bounding boxes is not expensive [15, 41]. We have found that many applications require accurate and expensive target DNNs, so we focus on reducing executing the target DNN wherever possible.

8 CONCLUSION

To reduce the cost of queries using expensive target DNNs, we introduce a method of constructing task-agnostic, trainable indexes for unstructured data. TASTI relies on the key property that many queries only require a low dimensional representation of unstructured data records. TASTI uses an embedding DNN and target DNN annotated cluster representatives as its index, which allows for more accurate and generalizable proxy scores across a range of query types. We theoretical 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. We hope that our work serves as a starting point for other indexing and query processing techniques over unstructured data.

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

[5] 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.

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

[7] 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.

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

[9] 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).

[10] 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.

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

[12] 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).

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

[14] 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.

[15] 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).

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

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

[18] 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.

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

[20] 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.

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

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

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

[24] 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).

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

[26] 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.

[27] 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.

[28] 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.

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

[30] 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.

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

[32] 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.

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

[34] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*.

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

[36] 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.

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

[38] 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).

[39] 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.

[40] 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.

[41] 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.

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