Boggart: Towards General-Purpose Acceleration of Retrospective Video Analytics

Neil Agarwal and Ravi Netravali, *Princeton University*

https://www.usenix.org/conference/nsdi23/presentation/agarwal-neil

This paper is included in the
Proceedings of the 20th USENIX Symposium on
Networked Systems Design and Implementation.

April 17–19, 2023 • Boston, MA, USA

978-1-939133-33-5

Open access to the Proceedings of the
20th USENIX Symposium on Networked
Systems Design and Implementation
is sponsored by

جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology

Boggart: Towards General-Purpose Acceleration of Retrospective Video Analytics

Neil Agarwal, Ravi Netravali
Princeton University

Abstract

Commercial retrospective video analytics platforms have increasingly adopted general interfaces to support the custom queries and convolutional neural networks (CNNs) that different applications require. However, existing optimizations were designed for settings where CNNs were platform- (not user-) determined, and fail to meet at least one of the following key platform goals when that condition is violated: reliable accuracy, low latency, and minimal wasted work.

We present Boggart, a system that simultaneously meets all three goals while supporting the generality that today's platforms seek. Prior to queries being issued, Boggart carefully employs traditional computer vision algorithms to generate indices that are imprecise, but are fundamentally comprehensive across different CNNs/queries. For each issued query, Boggart employs new techniques to quickly characterize the imprecision of its index, and sparingly run CNNs (and propagate results to other frames) in a way that bounds accuracy drops. Our results highlight that Boggart's improved generality comes at low cost, with speedups that match (and most often, exceed) prior, model-specific approaches.

1 Introduction

Video cameras are prevalent in our society, with massive deployments across major cities and organizations [4, 10, 11, 22, 32, 47]. These cameras continually collect video data that is queried *retrospectively* to guide traffic/city planning, business or sports analytics, healthcare, crime investigation, and many other applications [5, 14, 23, 26, 31, 33–35, 37, 61, 69, 122]. Queries typically involve running convolutional neural network (CNN) models that locate and characterize particular objects in scenes [53, 99, 104, 106, 125]. Applications tailor the architectures and weights of those CNNs to their unique requirements (e.g., accuracy, latency, and resource cost) and target tasks, e.g., via specialization to scenes or object types [8, 13, 116], proprietary training datasets [7, 27, 28].

To support these diverse applications, commercial video analytics platforms (e.g., Microsoft Rocket [41, 44, 45], Amazon Rekognition [39], Google AI [70], IBM Maximo [83]) have steadily transitioned away from exposing only predetermined video processing results, towards being platforms that allow users/applications to register custom, large-scale video analytics jobs without worrying about infrastructural details [55, 116, 118]. To register a query, users typically provide (1) a CNN model of arbitrary architecture and weights, (2) a target set of videos (e.g., feeds, time periods), and (3) an accuracy target indicating how closely the provided results must match those from running the CNN on every frame. Higher accuracy targets typically warrant more inference (and thus, slower responses and higher costs).

From a platform perspective, there exist three main goals for each registered query. First and foremost, provided results should *reliably* meet the specified accuracy target (usually above 80% [80, 92, 105, 116]). Subject to that constraint, the platform should aim to consume as few computational resources as possible (i.e., minimize unnecessary work) and deliver responses as quickly as possible. The main difficulty in achieving these goals stems from the potentially massive number of video frames to consider, and the high compute costs associated with running a CNN on each one. For example, recent object detectors would require 500 GPU-hours to process a week of 30-fps video from just one camera [77, 82].

Unfortunately, despite significant effort in optimizing retrospective video analytics [42, 48, 80, 81, 93–95], no existing solution is able to simultaneously meet the above goals for the general interfaces that commercial platforms now offer. Most notably, recent optimizations perform ahead-of-time processing of video data to build indices that can accelerate downstream queries [48, 80, 95]. However, these optimizations were designed for settings where models were known *a priori* (i.e., not provided by users), and thus deeply integrate knowledge of the specific CNN into their ahead of time processing. Porting these approaches to today's bring-your-own-model platforms fundamentally results in unacceptable accuracy violations and resource overheads. The underlying reason is that models with even minor discrepancies (in architecture or weights) can deliver wildly different results for the same tasks and frames. Consequently, using different models for ahead-of-time processing and user queries can yield accuracy drops of up to 94% (§2.3). Building an index for all potential models is unrealistic given the massive space of CNNs [102, 107, 114, 149], and the inherent risk of wasted resources since queries may never be issued [80, 137].

In this paper, we ask *"can retrospective video analytics platforms operate more like general-purpose accelerators to achieve their goals for the heterogeneous queries+models provided by users?"* We argue that they can, but doing so requires an end-to-end rethink of the way queries are executed, from the ahead-of-time processing used to develop indices, to the execution that occurs only once a user provides a model and accuracy target. We examine the challenges associated with each phase, and present **Boggart**, a complete video analytics platform that addresses those challenges.

Ahead-of-time processing (indexing). To support our goals, an index must meet the following criteria: (1) comprehensive with respect to data of interest for different models/queries – any information loss would result in unpredictable accuracy

drops, (2) links information across frames so CNN inference results – the most expensive part of query execution [80, 94] – can be propagated from one frame to another at low cost, and (3) cheap to construct since queries may never come in.

We show with Boggart that, if applied in a *conservative* manner, traditional computer vision (CV) algorithms [52, 88, 100, 127] can be repurposed to generate such an index per video. Along these lines, Boggart's ahead-of-time processing extracts a comprehensive set of *potential* objects (or blobs) in each frame as areas of motion relative to the background scene. Trajectories linking blobs across frames are then computed by tracking low-level, model-agnostic video features, e.g., SIFT keypoints [110]. Crucially, Boggart's trajectories are computed once per video (not per video/model/query tuple) using cheap CV tasks that require only CPUs, and are generated 58% faster than prior model-specific indices constructed using compressed CNNs and GPUs (§6.3).

Query execution. Once a user registers a query and CNN, the main question is how to use the comprehensive index to quickly generate results that meet the accuracy target, i.e., running inference on as few frames as possible, and aggressively propagating results along Boggart's trajectories. The challenge is that Boggart's index is extremely coarse and imprecise relative to CNN results. For instance, blob bounding boxes may be far larger than those generated by CNNs, and may include multiple objects that move in tandem. Worse, the imprecision of Boggart's index varies with respect to different models and queries; prior systems avoid this issue by using indices that directly approximate specific models.

To handle this, Boggart introduces a new execution paradigm that first selects frames for CNN inference in a manner that sufficiently bounds the potential propagation error from index imprecision and unavoidable inconsistencies in CNN results [97]. The core idea is that such errors are largely determined by model-agnostic features about the video (e.g., scene dynamics), and can be discerned via inference on only a small set of representative frames. CNN results are then propagated using a custom set of accuracy-aware techniques that are specific to each query type (e.g., detection, classification) and robustly handle (and dynamically correct) imprecisions in Boggart's trajectories.

Results. We evaluated Boggart using 96 hours of video from 8 diverse scenes, a variety of CNNs, accuracy targets, and objects of interest, and 3 widely-used query types: binary classification, counting, and detection. Across these scenarios, Boggart consistently meets accuracy targets while running CNNs on only 3-54% of frames. Perhaps more surprisingly given its focus on generality and model-agnostic indices, Boggart outperforms existing systems that (1) rely solely on optimizations at query execution time (NoScope [94]) by 19-97%, and (2) use model-specific indices (Focus [80] running with knowledge of the exact CNN) by -5-58%.

Taken together, our results affirmatively answer the question above, showing that Boggart can support the general interfaces and diverse user models that commercial platforms face, while delivering reliable accuracy and comparable (typically larger) speedups than prior, model-specific optimizations. The source code and experimental data for Boggart are available at https://github.com/neilsagarwal/boggart.

2 Background and Motivation

In this section, we first present an overview of retrospective video analytics pipelines and their use cases (§2.1). We then describe existing optimizations (§2.2), and present measurements highlighting their inability to generalize to the different models and queries that users register (§2.3). Additional related work can be found in §7.

2.1 Primer on Retrospective Video Analytics

Numerous applications leverage (and are guided by) insights gleaned from analyzing the large amount of video data previously captured in different environments. For example, sports analytics tools leverage video analytics on previous game film to detect players on a field; these detections are fed into tracking algorithms to determine the efficacy of various strategies and to evaluate player performance [16, 29]. Similarly, retail analysts use video analytics to locate customers in indoor environments with high accuracy, in order to understand customer-product interaction and, ultimately, to improve store layout designs and product placement [31, 34]. City planners and traffic engineers employ video analytics to extract trends from historical footage, e.g., identifying points of congestion or opportunities for expansion [3, 20, 33, 35].

Despite their diverse use cases, retrospective video analytics generally share two main properties that characterize their computational requirements. First, they typically process video frames using convolutional neural networks (CNNs), a class of deep neural networks that have become the norm for automated vision processing due to their success in extracting spatial dependencies within images [53, 99, 104, 106, 125]. CNNs incorporate 3 kinds of layers: convolutional (responsible for recognizing pixel-level features), pooling (responsible for making these features more abstract), and fully-connected (responsible for using acquired features for prediction). In a CNN, each successive layer learns a more complex feature representation. Earlier layers focus on simple features such as colors and edges, while later layers aim to recognize specific objects. We refer the reader to prior reports [80, 94, 103] for more details.

Second, retrospective video analytics applications typically use CNNs to perform *object-centric* queries, e.g., to locate, characterize, and label different types of objects in frames. Indeed, the output of a CNN is a set of bounding boxes that localize all identified objects in a given frame, with each box being accompanied by a probability distribution characterizing its potential labels (or types). Such object-centric queries subsume those reported by both recent academic literature [55, 64, 92, 94, 105] and industrial organizations that run video analytics platforms [36, 80, 87, 111,

116, 140]. Concretely, in this paper, we consider the following query types (and accuracy metrics):

- **binary classification**: return a binary decision as to whether a specific object or type of object appears in each frame. Accuracy is measured as the fraction of frames tagged with the correct binary value.
- **counting**: return the number of objects of a given type that appear in each frame. Per-frame accuracy is set to the percent difference between the returned and correct counts.
- **bounding box detection**: return the coordinates for the bounding boxes that encapsulate each instance of a specific object or object type. Per-frame accuracy is measured as the mAP score [67], which considers the overlap (IOU) of each returned bounding box with the correct one.

The heterogeneity in use cases also brings important differences between manifestations of retrospective video analytics applications. Most notably, applications often apply *specialized* CNNs that cater to their specific target environments, object(s) of interest, required accuracy, task complexity, and available computational resources [75, 80, 94, 113]. Recent analyses of production video analytics workloads have shown that applications carry out such specialization by (1) selecting an existing reference model architecture from a popular family (e.g., ResNet, YOLO) and (2) training that model using custom and/or proprietary datasets that yield desirable weights for the target use case [116].

2.2 Existing Acceleration Approaches

Query-time strategies. Systems such as NoScope [94] and Tahoma [42] only operate once a user issues a query. To accelerate response generation, they first train cascades of cheaper binary classification CNNs that are specialized to the user-provided CNN, object of interest, and target video. The specific cascade to use is selected with the goal of meeting the accuracy target while minimizing computation and data loading costs. If confidence is lacking with regards to meeting the accuracy target, the user's CNN is incrementally run on frames until sufficient confidence is achieved.

Ahead-of-time (preprocessing) strategies. Other systems provide speedups by performing some computation ahead of time, i.e., before a query is issued; for ease of exposition, we refer to such computations as *preprocessing* in the rest of the paper. For example, Focus [80] speeds up binary classification queries by building an approximate, high-recall index of object occurrences using a specialized and compressed CNN that roughly matches the full CNN on the target video. Objects are then clustered based on the features extracted by the compressed model such that, during query execution, the full CNN only runs on the centroid of each cluster, with labels being propagated to all other objects in the same cluster.

BlazeIt [93] and TASTI [95] accelerate aggregate versions of certain query types, e.g., total counts across all frames. Preprocessing for both systems involves generating sampled results using the full CNN. TASTI uses the sampled results to train a cheap embedding CNN that runs on all frames and clusters those that are similar from the model's perspective. During query execution, the full CNN is run only on select frames in each cluster, with the results propagated to the rest. In contrast, BlazeIt uses the sampled results to train specialized CNNs that act as control variates for the remaining frames: the specialized CNNs run on all frames, and the results are correlated with sampled results from the full CNN to provide guarantees in statistical confidence. OTIF [48] follows a similar strategy, but uses proxy models (trained using the sampled results) to extract tracks about model-specific objects that are later used to accelerate tracking queries.

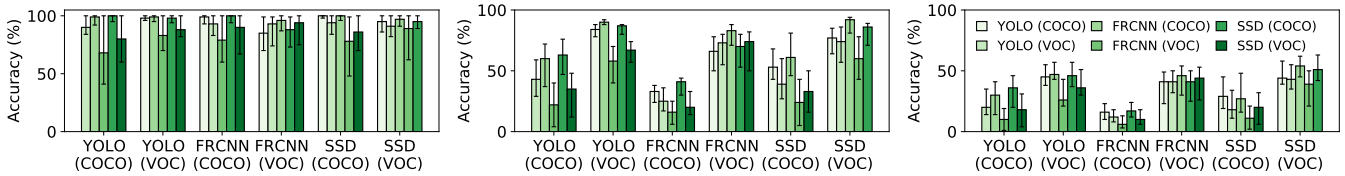
Videozilla [81] aims to extend such indexing optimizations across multiple video streams. More specifically, it identifies and exploits semantic similarities across streams that are based on the features extracted by the full CNN.

2.3 The Problem: Model-Specific Preprocessing

As confirmed by prior work [80, 93, 95] and our results in §6.3, preprocessing (intuitively) reduces the amount of computation required during query execution, and is crucial to enabling fast responses. However, *all* existing solutions suffer from the same fundamental issue: they deeply integrate a specific CNN into their preprocessing computations (e.g., to generate sampled results for training the compressed models used to build indices or group similarly-perceived frames), and assume that all future queries will use that same exact CNN. While such an approach was compatible with prior platforms that exposed only predetermined results from platform-selected CNN(s), it is no longer feasible with the bring-your-own-model interfaces that are now commonplace on commercial platforms. To make matters worse, consider that queries can be made at any point in the future and the space of potential CNNs is immense and rapidly evolving [102, 107, 114, 149], with variations in architecture (e.g., # of layers) or weights (e.g., different training datasets). In fact, building an index for even today's reference models would quickly present intractable resource challenges at the scale of retrospective video datasets: there exist tens of popular model families, each with multiple architecture options, e.g., the ResNet family alone has 8 architectures.

To quantify the issues when this assumption is violated, we ran experiments asking: how would accuracy be affected if the CNN provided by users during query execution (i.e., *query CNN*) was different than the CNN used during preprocessing (i.e., *preprocessing CNN*)? We consider the three query types above, videos and objects described in §6.1, and a wide range of CNNs: Faster RCNN, YOLOv3, and SSD, each trained on two datasets (COCO and VOC Pascal).

For each possible pair of preprocessing and query CNNs, we ran both CNNs on the video to obtain a list of object bounding boxes per frame. In line with Focus' observation that classification results from two CNNs may not identically match but should intersect for the top-k results [80], we ig-



(a) **Binary classification.** (b) **Counting.** (c) **Bounding box detection.**

Figure 1: **Query accuracies when *different* CNNs are used for preprocessing (bar types) and query execution (X axes). Bars show results for the median video, and error bars span the 25-75th percentiles. Models are listed as 'architecture (training dataset)'.**

nore the classifications from each CNN. Instead, we consider all bounding boxes from the preprocessing CNN that have an IOU of ≥ 0.5 with some box generated by the query CNN; results were largely unchanged for other IOU thresholds. This presents the *best scenario* (accuracy-wise) for existing preprocessing strategies. Finally, we compute query results separately using only the remaining preprocessing CNN's boxes or all of the query CNN's boxes, and compare them.

Figure 1 shows that discrepancies between preprocessing and query CNNs can lead to significant accuracy degradations, with the errors growing as query precision increases. For example, median degradations were 0-32% for binary classifications, but jump to 8-84% and 46-94% for counting and detections. Note that degradations for binary classification and counting are by definition due to the preprocessing CNN entirely missing objects relative to the query CNN. Parsed differently, median degradations across query types were 0-84%, 2-94%, and 1-90% when the preprocessing and query CNNs diverged in terms of only architecture, only weights, or both. Figure 2 shows that these degradations persist even for variants in the same family of CNNs.

Takeaway. Ultimately, when run on the general interfaces of today's commercial video analytics platforms where users can provide CNNs, all existing optimizations would sacrifice at least one key platform goal:

- *reliable accuracy:* running preprocessing optimizations as is (using platform-determined CNNs) would yield unpredictable and substantial (up to 94%) accuracy hits;
- *minimal wasted work:* performing preprocessing for all potential user CNNs is not only unrealistic given the sheer number of possibilities, but would also result in substantial wasted work since queries may never be issued;
- *low-latency responses:* optimizing only once a query is issued will yield higher than necessary response times.

3 Overview of Boggart

This section describes the overall workflow that Boggart uses to simultaneously meet all three platform goals for general, user-provided CNNs (Figure 3). §4 and §5 detail its preprocessing and query execution phases, and the project repository includes end-to-end visualizations of its operation [9].

Preprocessing. The main goal of Boggart's preprocessing phase is to perform cheap computations over a video dataset such that the outputs (an index) can accelerate query execution for diverse user CNNs, without sacrificing accuracy. Crucially, to avoid the pitfalls of prior work (§2.3), Boggart's preprocessing does not incorporate *any* knowledge of the specific CNN(s) that will be used during query execution. Instead, our insight is that traditional computer vision (CV) algorithms [88,100,124,127] are well-suited for such preprocessing, as they extract information purely about video data, rather than how a specific model or query would parse that data. Using generic CV algorithms enables Boggart to generate a single index per video, rather than per video/query/model tuple. Further, those CV algorithms are computationally cheaper than (even compressed) CNNs, and rely on CPUs (not GPUs), keeping monetary costs low (§6.3). Both aspects drastically reduce the potential for wasted work.

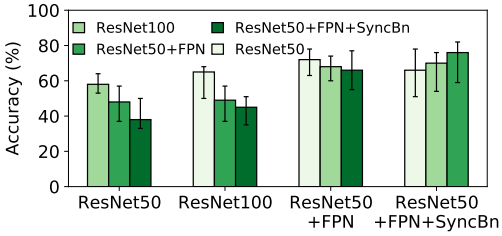


Figure 2: **Accuracies when CNNs for preprocessing (bar types) and query execution (X axis) are FasterRCNN+COCO with different ResNet backbones. Results are for counting queries; bars list medians with error bars for 25-75th percentiles.**

However, in contrast to their intended use cases, for our purposes, CV algorithms must be *conservatively* tuned to ensure that accuracy during query execution is not sacrificed. Namely, Boggart's index must *comprehensively* include all information that may influence or be incorporated in a query result (across CNNs), regardless of how coarse or imprecise that information is. Whereas coarse or imprecise results can be corrected or filtered out during query execution, missing information would result in unpredictable accuracy drops.

Accordingly, Boggart carefully uses a combination of motion extraction and low-level feature tracking techniques to identify all potential objects as areas of motion (or *blobs*) relative to a background estimate, and record their *trajectories* across frames by tracking each blob's defining pixels (or keypoints). For the former task, only high-confidence pixels are marked as being part of the background, ensuring that even minor motion is treated as a potential object; note that static objects are definitively discovered during query execution via CNN sampling on the frames across which the ob-

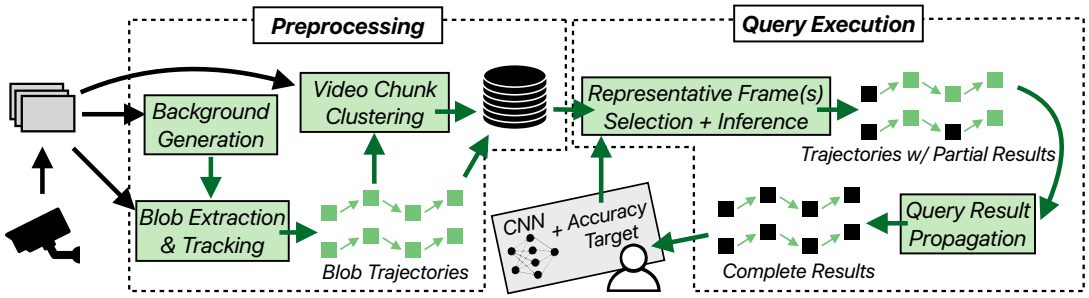


Figure 3: **Overview of Boggart.**

jects are static. For the latter task, any uncertainty in trajectory continuity (e.g., tracking ambiguities) is handled by simply starting a new trajectory; this ensures that results are not mistakenly propagated across different objects during query execution, albeit at the cost of additional inference. Overall, we did not observe *any* missed moving objects in Boggart's indices across our broad evaluation scenarios (§6.1).

Trajectories are a fundamental shift from the clustering strategies that prior systems use to group frames or objects based on how they are perceived by a specific CNN (§2.2). In contrast, trajectories are computed in a model-agnostic manner, but still provide a mechanism through which to propagate CNN results across frames during query execution – the primary source of speedups. Such generality does, however, come at a cost. Whereas prior efforts cluster on frames or object classes, Boggart's trajectories group frames on a per-object basis. This discrepancy lets Boggart defer the determination of how a user's CNN perceives each object to query execution, but it limits potential propagation, i.e., Boggart propagates the result for an object across the frames in which it appears, while prior approaches can propagate results across appearances of different objects. Note that this discrepancy does not apply to detection queries that require precise object locations (not just labels) to be propagated.

A natural question is: why not cluster objects on the features extracted by traditional CV algorithms to enable more result propagation? The issue is that, if performed without knowledge of the user-provided CNN, such clustering could lead to unsafe result propagation. More specifically, objects that are similar on some set of features but are perceived differently by the user's CNN could end up in the same cluster.

Query Execution. Once a user registers a query (providing a CNN, accuracy target, and video to consider), Boggart's goal is to generate a full set of per-frame results as quickly as possible, while reliably meeting the target accuracy. This translates to using the index from preprocessing (i.e., blobs and trajectories) to run the CNN on a small sample of frames, and efficiently propagate those results to the remaining frames.

The main challenge is that, owing to their general-purpose nature (relative to different models/queries) and closeness to noisy image signals, the CV algorithms used during preprocessing typically produce results that fail to precisely align with those from a user's CNN [55, 105]. Consequently, in being comprehensive, Boggart's index is coarse and imprecise relative to the target results from a user's CNN, e.g., with misaligned bounding boxes or extraneous objects that are not of interest to the query. Worse, the degree of imprecision is specific to the user CNN, and can lead to cascading errors (and accuracy drops) as results are propagated along Boggart's trajectories. All prior efforts avoid these issues by tuning indices to specific CNNs at the cost of generality.

To bound accuracy degradations (and reliably meet the specified target) while avoiding substantial inference, Boggart introduces a new query execution approach with two main components. First, to quickly and judiciously select the frames to run CNN inference on, our key observation is that errors from index imprecision and result propagation are largely dictated by model-agnostic features about the video, e.g., scene dynamics or trajectory lengths. Accordingly, Boggart clusters chunks of video in the dataset based on those features, and runs the user's CNN only on cluster centroids to determine the best frame selection strategy per cluster for the query at hand, i.e., the lowest frequency of CNN inference that meets the user-specified target accuracy. We note that, since clustering is based on model-agnostic features, it can be performed during preprocessing; CNN inference on centroids, however, only occurs once a user registers a query.

Second, to further limit inference overheads, Boggart introduces a new set of result propagation techniques that are specific to each query type and bolster propagation distances in spite of imprecisions in the index. For instance, for bounding box detections, Boggart leverages our empirical observation that the relative position between an object's keypoints (from preprocessing) and its bounding box edges remain stable over time. Building on this, Boggart propagates an object's CNN-produced bounding box to subsequent frames in its trajectory by efficiently searching for the coordinates that maximally preserve these spatial relationships.

Query model and assumptions. Boggart currently supports the large body of object-centric queries whose results are reported at the granularity of individual objects (e.g., labeling or locating them) and whose CNNs are run on a per-frame basis. Thus, currently handled queries include classifications, counting, and detections, as well as queries that build atop those primitives such as tracking and activity recognition. Such queries dominate the workloads reported

by commercial platforms [36, 80, 87, 116, 140], and subsume those supported by prior work (§2.2). We note that Boggart's approach is general enough to also accelerate less common, finer-grained queries, e.g., semantic segmentation [109]. For such queries, the keypoints (and their matches across frames) recorded in Boggart's index can be used to propagate groups of pixel labels; we leave implementing this to future work.

Boggart does not make any assumptions about or require any knowledge of the object type(s) that a query targets. Instead, as described above, Boggart relies on generic background estimation and motion extraction to identify potential objects. The intuition is that a moving object of any kind will involve (spatially correlated) moving pixels that can be identified purely based on the scene. Boggart leaves it to the user's CNN to determine whether those potential objects are of interest during query execution. We stress-test Boggart's robustness to different object types in §6.4.

Boggart's preprocessing operates on videos from static cameras that capture a single scene. Boggart currently does not support preprocessing for videos with changing backgrounds, e.g., CGI-generated films or videos from moving cameras. We note, however, that the CV community has actively been extending the core techniques that Boggart builds atop to deliver improved robustness in the face of moving cameras [65, 128, 135, 142]. We leave an exploration of integrating these efforts into Boggart to future work.

Reliance on Heuristics. Despite its focus on reliably meeting accuracy targets, Boggart's operation does involve multiple heuristics, i.e., tracking algorithms (§4), preset video chunk sizes (§4), thresholds for blob extraction (§4), and clustering parameters (§5.2). The upcoming sections and results in §6.4 elaborate on Boggart's sensitivity to each parameter. More generally, Boggart's approach to ensure sufficient accuracy is shared: each heuristic is conservatively configured to err on capturing too much data (resulting in unnecessary processing) rather than missing important data, i.e., prioritizing accuracy over efficiency. Examples include returning blobs for unlikely (but possible) objects, splitting trajectories upon uncertainty in object tracking, etc. §6 shows that this approach enables Boggart to consistently and efficiently deliver accurate query responses for diverse camera feeds, queries, models, objects, and accuracy targets.

4 Boggart's Preprocessing

Boggart's target output from preprocessing is a set of blobs and their trajectories. To efficiently extract this information and enable parallel processing over the dataset, Boggart operates independently on video chunks (i.e., groups of contiguous frames); the default chunk size is 1 min (profiled in §6.4), and trajectories are bound to individual chunks to eliminate any cross-chunk state sharing. The rest of this section describes the analysis that Boggart performs per chunk.

Background estimation. Extracting blobs inherently requires a point of reference against which to discern areas of motion. Thus, Boggart's first task is to generate an estimate of the background scene for the current chunk. However, existing background estimation approaches [46, 101] are ill-suited for Boggart as they are primarily concerned with generating a single, coherent background image despite scene dynamics (e.g., motion) that complicate perfect foreground-background separation. In contrast, Boggart's focus is on navigating the following tradeoff between accuracy and efficiency, not coherence. On one hand, placing truly background pixels in the foreground will lead to spurious trajectories (and query execution inefficiencies). On the other hand, incorrectly placing a temporarily static object in the background can result in accuracy degradations. Indeed, unlike entirely static objects that will surely be detected via CNN sampling and propagated to all frames in a chunk (during query execution), temporarily static objects may be missed and should only be propagated to select frames.

Boggart addresses the above tradeoff in a manner that favors accuracy. More specifically, Boggart only marks content as pertaining to the background scene when it has high confidence; all other content is conservatively marked as part of the foreground and is resolved during query execution. To realize this approach, Boggart eschews recent background estimation approaches in favor of a custom, lightweight strategy.

In its most basic form, background estimation involves recording the distribution of values assigned to each pixel (or region) across all frames in the chunk, and then marking the most frequently occurring value(s) (i.e., the peaks in the probability density function) as the background [124, 127]. This works well in scenarios where there is a clear peak in the distribution that accounts for most of the values, e.g., if objects do not pass through the pixel or do so with continuous motion, or if an object is entirely static and can thus be safely marked as the background. However, complications arise in settings with multiple peaks. For instance, consider a pixel with two peaks. Any combination of peaks could pertain to the background: a tree could sway back and forth (both), a single car could temporarily stop at a traffic light (one), or multiple cars could serially stop and go at the light (none).

To distinguish between these multi-modal cases and identify peaks that definitely pertain to the background for a chunk, Boggart extends (into the next chunk) the duration over which the distribution of pixel values is computed. The idea is that motion amongst background components should persist with more video, while cases with temporarily static objects should steadily transform into uni-modal distributions favoring either the background scene or the object (if it remains static). To distinguish between the object and background in the latter case, Boggart further extends the distribution of pixel values to incorporate video from the previous chunk. If the same peak continues to rise, it must pertain to the background since we know that the object was not static throughout the entire chunk. Otherwise, Boggart conservatively assigns an empty background for that pixel.

Figure 4: **Example screenshots from the Auburn video (Table 1). CNN (YOLOv3+COCO) detections are shown in white, while each of Boggart's trajectories (and their constituent blobs) is shown in a different color.**

Blob Extraction. Using the background estimate, Boggart takes a second pass through the chunk in order to extract areas of motion (blobs) on each frame. More specifically, Boggart segments each frame into a binary image whereby each pixel is annotated with a marker specifying whether it is in the foreground or background. Our implementation deems a pixel whose value falls within 5% of its counterpart(s) in the background estimate as a background pixel, but we find our results to be largely insensitive to this parameter. Given the noise in low-level pixel values [105], Boggart further refines the binary image using a series of morphological operations [115], e.g., to convert outliers in regions that are predominantly either background or foreground. Lastly, Boggart derives blobs by identifying components of connected foreground pixels [71], and assigning a bounding box using the top left and bottom right coordinates of each component.

Computing Trajectories. Boggart's final preprocessing task is to convert the set of per-frame blobs into trajectories that track each blob across the video chunk. At first glance, it may appear that sophisticated multi-object trackers (e.g., Kalman Filters) [50, 91, 134, 138] could directly perform this task. However, most existing trackers rely on pristine object detections as input. Blobs do not meet this criteria, and instead are far coarser and imprecise (Figure 4). At any time, a single blob may contain multiple objects, e.g., two people walking together. Blobs may split or merge as their constituent objects move and intersect. Lastly, the dimensions of a given object's blob bounding boxes can drastically fluctuate across frames based on interactions with the estimated background.

To handle these issues, we turn to tracking algorithms that incorporate low-level feature keypoints (SIFT [110] in particular) [88, 89], or pixels of potential interest in an image, e.g., the corners that *may* pertain to a car windshield. Associated with each keypoint is a descriptor that incorporates information about its surrounding region, and thus enables the keypoint (and its associated content) to be matched across images. Boggart conservatively applies this functionality to generate correspondences between blobs across frames.

For each pair of consecutive frames, Boggart pairs the constituent keypoints of each blob. This may yield any form of an N $\rightarrow$ N correspondence depending on the underlying tracking event, e.g., blobs entering/leaving a scene, fusion or splitting of blobs. For instance, if the keypoints in a blob on frame f_i match with keypoints in N different blobs on frame f_{i+1}, there is a 1 $\rightarrow$ N correspondence. To generate trajectories, Boggart makes a series of forwards and backwards scans through the chunk. For each correspondence that is not 1 $\rightarrow$ 1, Boggart propagates that information backwards to account for the observed merging or splitting. For example, for a 1 $\rightarrow$ N correspondence between frames f_i and f_{i+1}, Boggart would split f_i's blob into N components using the relative positions of the matched keypoints on f_{i+1} as a guide.

Index Storage. Preprocessing outputs are stored in MongoDB [1]; overheads are profiled in §6.4. Matched keypoints are stored with the corresponding frame IDs: row = [<((x,y)-coordinates, frame #)>]. Blob coordinates (and their trajectory IDs) are stored per frame to facilitate the matching of CNN results and blobs on sampled frames during query execution (§5.1): row = [<((x,y)-coordinates of top left corner, (x,y)-coordinates of bottom right corner, trajectory ID)>].

5 Fast, Accurate Query Execution

During query execution, Boggart's sole goal is to judiciously use the user-provided CNN and the index from preprocessing to quickly generate a complete set of results that meet the specified accuracy target. Doing so involves answering two questions: (1) what sampled (or *representative*) frames should the CNN be run on such that we can sufficiently adapt to the registered query (i.e., CNN, query type, and accuracy target) and bound errors from index imprecisions?, and (2) how can we use preprocessing outputs to accurately propagate sampled CNN results across frames for different query types? For ease of exposition, we describe (2) first, assuming CNN results on representative frames are already collected.

5.1 Propagating CNN Results

Regardless of the query type, Boggart's first task is to pair the CNN's bounding box detections on representative frames with the blobs on those same frames; this, in turn, associates detections with trajectories, and enables cross-frame result propagation. To do this, we pair each detection bounding box with the blob that exhibits the maximum, non-zero intersection. Trajectories that are not assigned to any detection are deemed spurious and are discarded. Further, detections with no matching blobs are marked as 'entirely static objects' and are handled after all other result propagation (described below). Note that, with this approach and in spite of the trajectory corrections from §4, multiple detections could be associated to a single blob, i.e., when objects move together and

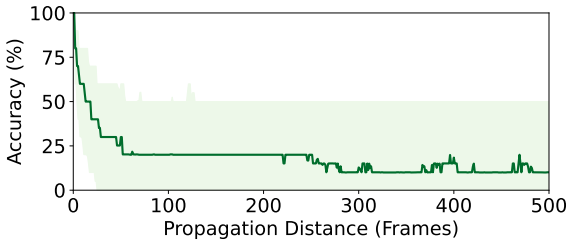


Figure 5: **Accuracy (mAP) degradations when CNN bounding boxes are propagated by computing the blob-to-detection coordinate transformation on a representative frame, and applying it to all other blobs in the trajectory. Line represents median detections, with ribbons spanning 25-75th percentiles.**

never separate. Using these associations, Boggart propagates CNN results via techniques specific to the target query type.

Binary classification and counting. To support both query types, each trajectory is labeled with an object count according to the number of detections associated with it on representative frames. If a trajectory passes through multiple representative frames, Boggart partitions the trajectory into segments, and assigns each segment a count based on the associations from the closest representative frame. Lastly, Boggart sums the counts across the trajectories that pass through each frame, and returns either the raw count (for counting), or a boolean indicating if count > 0 (for binary classification).

Bounding box detections. Whereas binary classification and count queries simply require propagating coarse information about object presence, bounding box queries require precise positional information to be shared across frames. However, as noted in §4, blobs and trajectories are inherently imprecise and fail to perfectly align with detections. A natural approach to addressing such discrepancies is to compute coordinate transformations between paired detections and blobs on representative frames, and apply those transformations to the remainder of each blob's trajectory; equivalently, one could compute transformations for a blob across its own trajectory, and apply them to add detections to non-representative frames. Unfortunately, Figure 5 shows that detection accuracy rapidly degrades with this approach, e.g., median degradations are 30% when propagating a box over 30 frames. The reason is that blobs and their paired detections move/resize differently across frames, resulting in median errors of 84% between the Euclidean distances of blob-blob and detection-detection coordinate transformations.

To fill the void of stable propagation mechanisms, Boggart leverages our finding that the relative positions between an object's constituent keypoints (i.e., those extracted and tracked during preprocessing) and its detection bounding box edges remain largely unchanged over short durations; we refer to these relative positions as *anchor ratios* since they 'anchor' an object's content to a relative position within the bounding box. This stability is illustrated in Figure 6, and is intuitive: objects tend to remain rigid over short time scales, implying that the points they are composed of move in much

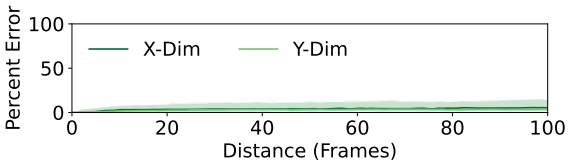


Figure 6: **Percent difference in anchor ratios for each object's keypoints across its trajectory. Lines show medians, with ribbons spanning 25-75th percentiles.**

the same way as the entire object does (including as the object scales in size). Building on this, Boggart propagates detections by using matching keypoints along the trajectories to which they have been associated, and efficiently solving an optimization problem in search of bounding box coordinates that maximally preserve the anchor ratios for each keypoint.

More formally, for each detection on each representative frame, Boggart considers the set of keypoints K that fall in the intersection with the associated blob. Each keypoint k in K has coordinates (x_k, y_k). Further, let the coordinates of the detection bounding box be (x_1, y_1, x_2, y_2), where (x_1, y_1) and (x_2, y_2) refer to the top left and bottom right corners. The anchor ratios (ax_k, ay_k) for keypoint k are computed as:

$$(ax_k, ay_k) = \left(\frac{x_2 - x_k}{x_2 - x_1}, \frac{y_2 - y_k}{y_2 - y_1}\right) \quad (1)$$

For each subsequent non-representative frame (until the next representative frame) that includes the same trajectory, Boggart finds the set of keypoints that match with those in K; denote the set of matching keypoints as K', where each k' in K' matches with keypoint k in K. Finally, to place the bounding box on the subsequent frame, Boggart solves for the corresponding coordinates (x_1, y_1, x_2, y_2) by minimizing the following function to maximally preserve anchor ratios:

$$\sum_{k'}^{K'} \left[\left(\frac{x_2 - x_{k'}}{x_2 - x_1} - ax_k \right)^2 + \left(\frac{y_2 - y_{k'}}{y_2 - y_1} - ay_k \right)^2 \right] \quad (2)$$

Note that this optimization (which takes 1 ms for the median detection) can be performed in parallel across frames and across detections on the same frame. Further, Boggart initializes each search with the coordinates of the corresponding detection box on the representative frame, thereby reducing the number of steps to reach a minima.

Propagating entirely static objects. Thus far, we have only discussed how to propagate detection bounding boxes that map to a blob/trajectory, i.e., moving objects. However, recall from §4 that certain objects which are entirely static will be folded into the background. These objects are discovered by the CNN on representative frames, but they will not be paired with any blob. Instead, Boggart broadcasts these objects to nearby frames (until the next representative frame) in a query-specific manner: such objects add to the per-frame counts used for classification and count queries, and their boxes are statically added into frames for detection queries.

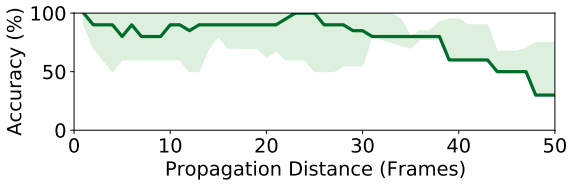


Figure 7: **Accuracy (mAP) degradations grow as Boggart propagates detection bounding boxes over longer durations. Results consider all object trajectories in the median video. Line represents medians, with ribbons spanning 25-75th percentiles.**

5.2 Selecting Representative Frames

To use the result propagation techniques from §5.1, we must determine the set of sampled, representative frames to collect CNN results on. Because CNN execution is the largest contributor to query execution delays (§6.4), we aim to select the smallest set of representative frames such that Boggart can sufficiently discern the relationship between its index and the CNN results, and generate a complete set of accurate results.

A natural strategy for selecting representative frames is to pick the smallest set of frames such that every trajectory appears at least once. In theory, executing the CNN on this set of frames should be sufficient to generate a result (e.g., object label, bounding box) for each trajectory, and propagate that result to all of the trajectory's frames. However, this straightforward approach falls short for two reasons:

1. CNNs can be inconsistent and occasionally produce different results for the same object across frames, e.g., a car in frame i may be ignored in frame $i+1$ [97, 98]. In line with prior analyses, we mostly observe this behavior for small or distant objects, e.g., YOLOv3 mAP scores are 18% and 42% for the small and large objects in the COCO dataset [120]. The consequence is that, if such an inconsistent result appears on a representative frame, Boggart would propagate it to all other frames in the trajectory, thereby spreading the error.
2. Even for consistent CNN results, propagation errors inherently grow with longer trajectories (i.e., as a given result is propagated to more frames). For instance, median accuracies are 90% and 30% when Boggart propagates bounding boxes over 10 and 50 frames (Figure 7).

These issues are more pronounced in busy/dynamic scenes with significant object occlusions/overlap [79, 132]. Moreover, the implication of both is that solely ensuring that the set of representative frames covers each trajectory is insufficient and can result in unacceptable accuracy degradations. To address this, Boggart introduces an additional constraint to the selection of representative frames: any blob in a trajectory must be within *max_distance* frames of a representative frame that contains the same trajectory. This, in turn, bounds both the duration over which inconsistent CNN results can be propagated, as well as the magnitude of propagation errors.

Tying back to our original goal, we seek the largest *max_distance* (and thus, fewest representative frames) that allows Boggart to meet the accuracy target. However, the appropriate *max_distance* depends on how the above issues manifest with the current query, CNN, and video. Digging deeper, we require an understanding of how Boggart's propagation techniques (for the query type at hand) and the user's CNN interact with each frame and trajectory, i.e., how accurate are Boggart's propagated results compared to the CNN's results. Though important for ensuring sufficient accuracy, collecting this data (particularly CNN results) for each frame during query execution would forego Boggart's speedups.

To achieve both accuracy and efficiency, Boggart clusters video chunks based on properties of the video and its index that characterize the aforementioned issues. The idea is that the chunks in each resulting cluster should exhibit similar interactions with the CNN and Boggart's result propagation, and thus should require similar *max_distance* values. Accordingly, Boggart could determine the appropriate *max_distance* for all chunks in a cluster by running the CNN and result propagation only on the cluster's centroid chunk.

To realize this approach, for each chunk, Boggart extracts distributions of the following features: object sizes (i.e., pixel area per blob), trajectory lengths (i.e., number of frames), and busyness (i.e., number of blobs per frame and trajectory intersections). These match our observations above: CNN inconsistencies are most abundant in frames with small objects, the potential for propagation errors is largest with long trajectories, and both issues are exacerbated in busy scenes.

With these features, Boggart clusters chunks using the K-means algorithm. We find that setting the number of target clusters to ensure that the centroids cover 2% of video strikes the best balance between CNN overheads and robustness to diverse and rapidly-changing video chunks; we profile this parameter in §6.4. Note that since clustering is based on model-agnostic features (from the extracted trajectories), it can be performed during preprocessing. Then, during query-execution, for each resulting cluster, Boggart runs the CNN on all frames in the centroid chunk. Using the collected results, Boggart runs its result propagation for a range of possible *max_distance* values, and computes an achieved accuracy for each one relative to the ground truth CNN results. More precisely, for each *max_distance*, Boggart selects the set of representative frames by greedily adding frames until our criteria is met, i.e., all blobs are within *max_distance* of the closest representative frame containing the same trajectory. From there, Boggart selects the largest *max_distance* that meets the specified accuracy goal, and applies it to pick representative frames for all other chunks in the same cluster.

Figure 8 highlights the effectiveness of Boggart's clustering strategy in terms of (quickly) adapting to different query types, accuracy targets, objects of interest, and CNNs. As shown in Figure 8(top), the median discrepancy between each chunk's ideal *max_distance* value and that of the corresponding cluster centroid is only 0-8 frames; this jumps to 45-898 frames when comparing chunks with the centroid

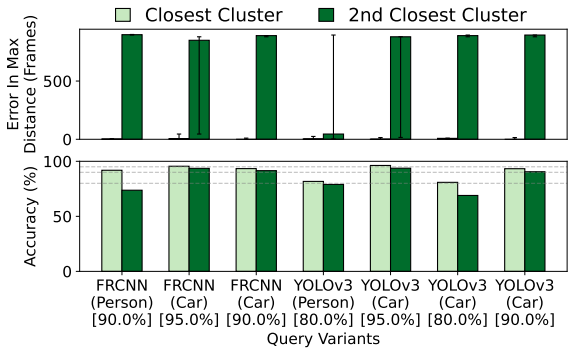


Figure 8: **Effectiveness of Boggart's clustering with different CNNs, (object types), and [accuracy targets]. Results are for the median video, and compare the ideal *max_distance* value for each chunk with those of the centroids in its cluster and the nearest neighboring cluster. The top graph measures the discrepancies in per-chunk *max_distance* (bars list medians, with error bars for 25-75th percentile); the bottom graph evaluates the corresponding hits on average accuracy (for detections).**

of the closest neighboring cluster. Figure 8(bottom) illustrates the importance of shrinking these discrepancies. More specifically, applying each centroid's ideal *max_distance* to all chunks in the corresponding cluster (i.e., Boggart's approach) yields average accuracies that are consistently above the targets. The same is not true when using the ideal *max_distance* values from the nearest neighboring cluster.

In summary, Boggart meets accuracy targets through co-analysis of the video content and specified query, i.e., object of interest, model, and query type. Boggart performs query/model-specific profiling of representative video chunks (where representative is defined by video content/-dynamics) to identify the frame inference strategy that most efficiently meets the accuracy target, and then executes this strategy for the remaining video chunks within each cluster.

6 Evaluation

We evaluated Boggart on a wide range of queries, CNNs, accuracy targets, and videos. Our key findings are:

- Boggart consistently meets accuracy targets while running the CNN on only 3-54% of frames, highlighting its comprehensive (model-agnostic) index and effective adaptation during query execution.
- Despite its goal of generality, Boggart's response times are 19-97% lower than NoScope's. Compared to Focus (which requires a priori knowledge of the CNN that will be used during query execution), Boggart's response times are 33% and 52% lower on counting and detection queries, and only 5% higher on classifications.
- Boggart's preprocessing (and index construction) runs 58% faster than Focus', while also generalizing to different CNNs/queries and requiring only CPUs (not GPUs).
- Boggart's preprocessing and query execution tasks speed up nearly linearly with increasing compute resources.

Camera location	Resolution
Auburn, AL (University crosswalk + intersection) [12]	1920×1080
Atlantic City, NJ (Boardwalk) [24]	1920×1080
Jackson Hole, WY (Crosswalk + intersection) [17]	1920×1080
Lausanne, CH (Street + sidewalk) [18]	1280×720
Calgary, CA (Street + sidewalk) [2]	1280×720
South Hampton, NY (Shopping village) [15]	1920×1080
Oxford, UK (Street + sidewalk) [21]	1920×1080
South Hampton, NY (Traffic intersection) [25]	1920×1080

Table 1: **Summary of our main video dataset.**

6.1 Methodology

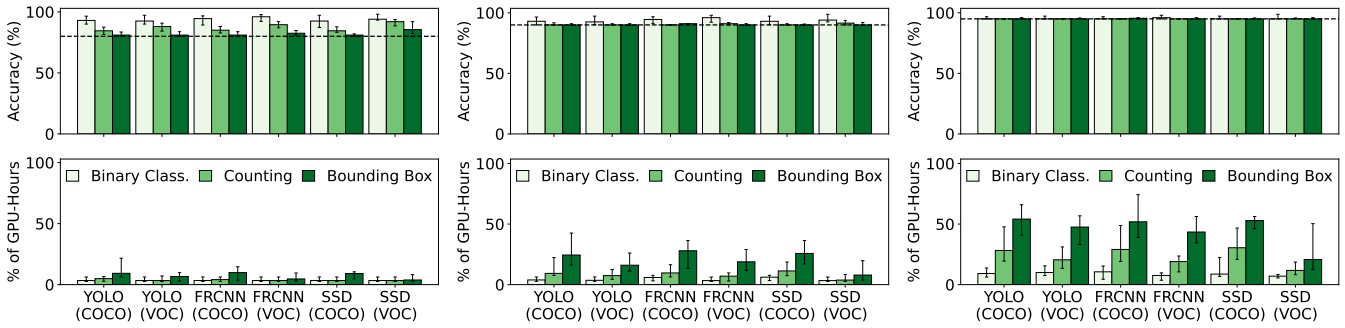
Videos. Table 1 summarizes the primary video sources used to evaluate Boggart. Video content across the cameras exhibits diversity in setting, resolution, and camera orientation (relative to the scene). From each camera, we scraped 12 hours of continuous video (at 30 fps) in order to capture varying levels of lighting and object densities (i.e., busyness). We consider additional videos and scene types in §6.4.

Queries. We consider the three query types (and their corresponding accuracy definitions) described in §2, i.e., binary classification, counting, and bounding box detection. For each type, we ran the query across our entire video dataset, and considered two objects of interest, people and cars, that cover drastically different size, motion, and rigidity properties; §6.4 presents results for additional object types. We evaluated Boggart with three accuracy targets – 80%, 90%, and 95% – and report accuracies as averages for each video. Accuracies are computed relative to running the model directly on all frames; as in prior systems and commercial platforms [41, 64, 80, 87, 105], Boggart does not aim to improve the accuracy of the provided model, and instead targets the same per-frame results at lower resource costs and delays.

CNN models. We consider three popular architectures: (1) SSD with a ResNet-50 backbone, (2) Faster RCNN with a ResNet-50 backbone, and (3) YOLOv3 with a Darknet53 backbone. For each, we used one version trained on the COCO dataset, and another trained on VOC Pascal. Trends for any results shown on a subset of CNNs (due to space constraints) hold for all considered models.

Hardware. Experiments used a server with an NVIDIA GTX 1080 GPU (8 GB RAM) and 18-core Intel Xeon 5220 CPU (2.20 GHz; 125 GB RAM), running Ubuntu 18.04.3.

Metrics. In addition to accuracy, we evaluate query execution performance of all considered systems (Boggart, Focus [80], and NoScope [94]) in terms of the number of GPU-hours required to generate results. We report GPU-hours for two reasons: (1) CNN execution (on GPUs) accounts for almost all response generation delays with all three systems, and (2) it is directly applicable to all of the systems, e.g., it incorporates NoScope's specialized CNNs. For preprocessing, we report both GPU- and CPU-hours since Boggart only requires the latter. As in prior work [80, 94], we exclude the video decoding costs shared by all considered systems.



(a) **80% Accuracy Target** (b) **90% Accuracy Target** (c) **95% Accuracy Target**

Figure 9: **Boggart's query execution performance across CNNs, query types, and accuracy targets; results are aggregated across object types. Bars summarize the distributions of per-video average result accuracy (top) and percentage of GPU-hours required to generate results relative to running the CNN on all frames (bottom). Bars list medians with error bars spanning 25-75th percentiles.**

Object Type →	People		Cars	
Query Type ↓	Acc.	% GPU-hrs	Acc.	% GPU-hrs
Binary Classif.	92%	6%	98%	3%
Counting	90%	11%	90%	7%
Bounding Box	91%	27%	90%	16%

Table 2: **Average accuracy and percentage of GPU-hours (relative to the naive baseline) for different query types and objects of interest. Results list median per-video values across all CNNs.**

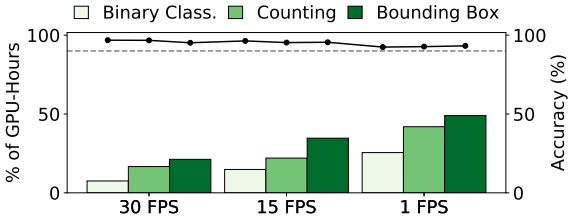


Figure 10: **Average accuracy (line) and percentage of GPU hours (relative to the naive baseline) for different video sampling rates. Results are listed for the median video, and consider YOLOv3+COCO and a 90% accuracy target.**

6.2 Query Execution Speedups

Figure 9 evaluates Boggart's query response times relative to a naive baseline that runs the CNN on all frames. Boggart always used the same, model-agnostic index per video.

There are three points to take away from Figure 9. First, across all of the conditions, Boggart *consistently* meets the specified accuracy targets. Second, the percentage of GPU-hours required to meet each accuracy target with Boggart grows as we move from coarse classification and counting queries to finer-grained bounding box detections. For example, with a target accuracy of 90%, the median percentage of GPU-hours across all models was 3-6%, 4-11%, and 8-28% for the three query types, respectively. Third, the percentage of GPU-hours also grows as the target accuracy increases for each query type. For instance, for counting queries, the percentage (across all CNNs) was 3-5% when the target accuracy was 80%; this jumps to 12-30% when the target accuracy grows to 95%. The reason is intuitive: higher accuracy targets imply that Boggart must more tightly bound the duration over which results are propagated (to limit propagation errors) by running the CNN on more frames.

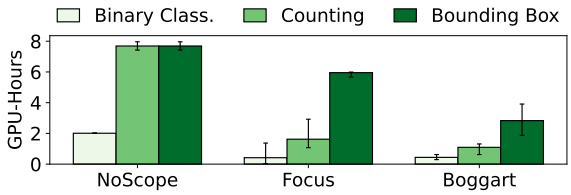
Different object types. Table 2 reports the results from Figure 9 separately per object type. As shown, the high-level trends from above persist for each. However, for a given query type, the percentage of required GPU-hours is consistently lower when considering cars versus people. The reason is twofold. First, inconsistencies in CNN results are more prevalent for people since they appear as smaller objects in our scenes (§5.2). Second, cars are inherently more rigid than people, and thus deliver more stability in the anchor ratios that Boggart relies on for bounding box propagation (§5.1); consequently, propagation errors for bounding box queries grow more quickly with people. Boggart handles both issues by running the CNN on more representative frames.

Downsampled video. Users may issue queries on sampled versions of each video [80]. We evaluated Boggart with three different sample rates: {30, 15, 1} fps. Although the number of considered frames drops, Figure 10 shows that Boggart's query execution speedups persist when operating over downsampled videos. For instance, with 1-fps video, Boggart requires only 25-49% of the GPU-hours that the naive baseline would need across all query types. Figure 10 also shows that Boggart's ability to consistently meet accuracy targets holds across sampling rates. We find that Boggart can hit accuracy targets without resorting to running the CNN on all frames because object keypoints – the primitive that Boggart tracks across frames during both trajectory construction (preprocessing) and detection propagation (query execution) – typically persist across frames even at these sample rates. For instance, Boggart matches 85% of the median object's keypoints across the 29-frame gap induced by the 1-fps rate.

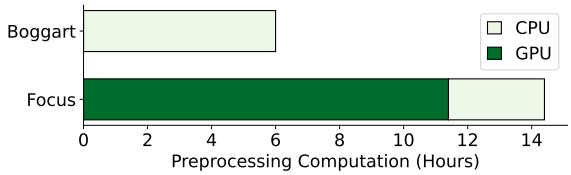
6.3 Comparison to State-of-the-Art

We compared Boggart with two recent retrospective video analytics systems: (1) NoScope [94], which only employs optimizations during query execution, and (2) Focus [80], which performs model-specific preprocessing by assuming a priori knowledge of user CNNs. §2.2 details each system.

For these experiments, we set the user-provided CNN to be YOLOv3+COCO, and the accuracy target to be 90%. Our



(a) **Query execution efficiency. Bars list values for the median video, with error bars spanning 25-75th percentiles.**



(b) **Preprocessing efficiency. Bars list GPU/CPU-hours for the median video. Note that NoScope does not perform preprocessing.**

Figure 11: **Comparing Boggart, Focus [80], and NoScope [94]. Results use YOLOv3+COCO and a target accuracy of 90%.**

Focus implementation used Tiny YOLO [120] as the specialized/compressed model (i.e., we ran Focus as if it knew the user CNN a priori), while NoScope used all of its open-source models. Following the training methodology used in both papers, we train the specialized/compressed models on 1-fps versions of the first half (i.e., 6 hours) of each video in our dataset, and run queries on the second half of each video.

Query Execution. Figure 11a compares the query response times of all three systems. As shown, Focus requires 5% fewer median GPU-hours than Boggart for binary classification queries. The main reason is that Focus' model-specific preprocessing (i.e., clustering of objects) enables more result propagation than Boggart's general, model-agnostic trajectories, i.e., Focus can propagate labels across objects, whereas Boggart can propagate labels only along a given object's trajectory (§3). Median propagation distances for results from the full CNN are 58 and 44 frames with Focus and Boggart.

Summing Focus' classifications to generate per-frame counts was insufficient for our 90% target. Thus, for counting queries, we performed favorable sampling until Focus hit 90% in each video: we greedily select a set of contiguous frames with constant count errors, run the CNN on a single frame, and correct errors on the remaining ones in the set. Even with such favorable sampling, Boggart required 33% fewer GPU-hours than Focus for counting queries.

Bounding box detections paint a starker contrast, with Boggart needing 52% fewer GPU-hours than Focus. Unlike with classification labels, Focus cannot propagate bounding boxes across frames. Instead, to accelerate these queries, Focus relies on binary classification, and runs the full CNN on all frames with an object of interest (to obtain their bounding boxes); for our videos, this translates to running the full CNN on 63-100% of frames. In contrast, Boggart propagates bounding boxes along each trajectory (median propagation distance of 23 frames) and reduces CNN tasks accordingly.

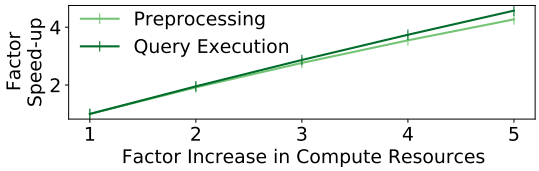


Figure 12: **Boggart's performance with increasing compute resources. Resource factors are multiples of the 18-core CPU and single GPU listed in §6.1. Results consider YOLOv3+COCO, a 90% accuracy target, and the median video.**

Compared to NoScope, Boggart's query execution tasks consume 19-97% fewer GPU-hours across query types. Boggart's speedups are largely due to three limitations with NoScope. First, NoScope does not perform preprocessing, and instead must train and run inference with its specialized and compressed CNNs during query execution. Second, results are not propagated across frames. Third, bounding box detections are sped up only via binary classification; note that NoScope performs binary classification on each frame (not object, like Focus), so we cannot simply sum the classification results to answer a counting query, and instead must execute counting queries as bounding box queries.

Preprocessing. Figure 11b shows that Boggart's preprocessing tasks take 58% fewer computation hours than Focus'. The discrepancy is from the training and inference costs that Focus incurs by using a specialized/compressed model. Note that *all* of Boggart's preprocessing is CPU-based, while Focus' costs are dominated (79%) by GPU operations. Further, Boggart's preprocessing runs once per video to support all future CNNs. To avoid accuracy drops (§2.3), Focus would have to run preprocessing for each CNN it wishes to support, leading to higher costs and potential for wasted work.

6.4 Profiling Boggart

Dissecting Boggart's performance. Boggart's preprocessing delays are dominated (83% on the median video) by the extraction of SIFT keypoints across frames; background estimation, trajectory construction, and clustering together account for only 17% of the time. Query execution profiles are similar, with CNN inference on centroid chunks and representative frames contributing 7% and 91% of runtime; result propagation (mostly for detections) takes the remaining 2%.

Resource scaling. Figure 12 shows that Boggart's preprocessing and query execution performance scale nearly linearly with increasing CPU and GPU resources, respectively. The reason is that feature extraction and CNN inference, the tasks that dominate delays in the two phases, inherently operate on a per-frame basis and can thus naturally be parallelized across frames. Note that these results only consider parallel processing within each chunk; Boggart can also parallelize across chunks since trajectories are bound to single chunks, i.e., there is no cross-chunk state sharing (§5).

Storage costs. Boggart's preprocessing generates, on average, 306 MB of data per 1 hour of video. For context, (1) the

average video in our dataset consumes 1 GB when encoded with H.264, and (2) Focus' preprocessing generates 70 MB of data for the same video. Recall that NoScope does not involve video preprocessing, and thus does not incur storage costs for indices. Note that 98% of Boggart's storage overheads are for keypoints used to propagate bounding boxes; blobs and trajectories consume only 2%.

Sensitivity to parameters. Boggart includes parameters for video chunk size (default: 1 min) and target number of clusters (default: centroids cover 2% of video). On average, we find that Boggart's performance is largely insensitive to both: varying chunk sizes from 0.2-10 min and the videos covered by centroids from 0.5-5% altered Boggart's performance by less than 5% (note that accuracy never dropped below the targets). However, the effects of each parameter are more pronounced on short amounts of video and are dependent on the content being considered. More specifically, smaller chunk sizes reduce the potential result propagation, but also shrink cluster centroids and increase the potential for parallel processing. Similarly, more clusters implies fewer suboptimalities in the selection of representative frames, but also additional centroids on which to run the CNN.

Generalizability. To further evaluate Boggart's ability to generalize, we ran experiments with three additional videos (3 hours each) and new object types specific to those scenes: birds in nature [19], boats in a canal [30], and people, cups, chairs, and tables in a restaurant [6]. For these experiments, we ran Boggart in the same way as above, i.e., it is not tuned in any way to the video or objects of interest. We also ran experiments considering different object types (trucks and bicycles) in the traffic videos from Table 1; these experiments used the same indices as in our main evaluation. All results exhibit similar trends as above, with Boggart always meeting accuracy targets (80%, 90%, 95%) and running CNNs on only 11.7-34.2%, 11.7-53.4%, and 12.6-56.7% of frames for binary classification, counting, and detection.

7 Additional Related Work

Live video analytics. Multiple systems accelerate queries on live video, with optimizations along the following axes: (1) profiling pipeline knobs to identify cheaper (but accurate) configurations [87, 140], (2) integrating on-camera or edge server resources for partial inference, frame filtering, or reusing results from prior frames [43,54,58,63,66,73,74, 105,129,136,141,148], (3) content/model-aware encoding to reduce data transfers [64, 133], and (4) spatiotemporal coordination for efficient multi-camera queries [86, 111]. These systems target an entirely different computational model (stream processing vs. "after-the-fact" querying) and thus face a different set of goals, optimization knobs, and constraints, e.g., by not having the entire dataset up front, live analytics can only propagate results to later frames.

Accelerating GPU tasks. One line of work optimizes DNNs for accelerated inference via distillation [78], quantization [60, 84, 144], or pruning [51, 108]. Another direction targets faster inference for a model, either through better scheduling of GPU resources across inference tasks [85,123, 126], or hardware acceleration [38,68,90,117]. These works are complementary to Boggart, which focuses on reducing the number of frames on which inference must be performed.

Video Object Detection. In addition to those in §4, Boggart builds on a line of work in the CV community that leverages the spatiotemporal aspect of video to accelerate detection and classification tasks. These techniques swap inference on sampled frames with optical flow networks that extend results from earlier frames [49, 56, 57, 59, 62, 72, 76, 98, 112, 130,131,145–147], and are thus similar in spirit to Boggart's result propagation strategy. However, unlike Boggart, these approaches are model-specific, in that the networks used for propagation must be trained according to the specific CNN (e.g., its feature extractor) used in the target query.

Video storage and indexing. Many systems balance video storage and lookup costs for specific query types [121, 139, 143] or CNNs [40, 96, 119, 137]. Boggart is complementary to these works in that its focus is on performing generalizable preprocessing and accelerating response generation after video frames are loaded into memory.

8 Conclusion

This paper described Boggart, a system for retrospective video analytics that supports the general "bring your own model" interfaces that are now commonplace in commercial platforms. To meet the core accuracy, speed, and efficiency goals of those platforms, Boggart holistically rethinks the query execution process, introducing cheap techniques to generate comprehensive (but imprecise) indices during preprocessing, and later use those indices to limit costly inference while bounding accuracy drops from imprecisions. Our results show that such generality can come at low cost, as Boggart outperforms prior, model-specific approaches.

Ethics. The focus of this work is on making the ethical processing of videos (public or private, according to the law) more efficient. We do not advocate for the processing of video for illicit purposes, unlawful tracking, etc. Moreover, Boggart is developed to improve the resource efficiency of existing retrospective video analytics platforms in a manner that does not change the interfaces they expose, i.e., the videos, models/queries, and customers they handle remain unchanged. In sum, Boggart does not alter the set of information exposed to applications – the videos that an application can query and the queries that the application can run on those videos are unchanged, and Boggart's internal state (e.g., preprocessing results) is not exposed.

Acknowledgements. We thank Ganesh Ananthanarayanan, Amit Levy, Jennifer Rexford, the NSDI reviewers, and our shepherd, Siddhartha Sen, for their valuable feedback and constructive comments. This work was supported by a Sloan Research Fellowship, a Cisco grant, and NSF CNS grants 2152313, 2153449, 2147909, and 2140552.

REFERENCES

[1] https://www.mongodb.com/.

[2] 11 Street SW Calgary. https://www.youtube.com/watch?v=iGxFLjqhkSA.

[3] 3 Reasons Why City Planners Need Video Analytics. https://www.briefcam.com/resources/blog/3-reasons-why-city-planners-need-video-analytics/.

[4] Absolutely everywhere in beijing is now covered by police video surveillance. https://qz.com/518874/.

[5] Are we ready for ai-powered security cameras? https://thenewstack.io/are-we-ready-for-ai-powered-security-cameras/.

[6] Beach Bar St. John Webcam. https://www.youtube.com/watch?v=2wqpy036z24.

[7] Betterview Combines Computer Vision and Post-Event Imagery to Map Tornado Damage. https://blog.betterview.com/betterview-combines-computer-vision-and-post-event-imagery-to-quickly-map-tornado-damage.

[8] Bird Cams Lab. https://www.zooniverse.org/organizations/cornellbirdcams/bird-cams-lab.

[9] Boggart Repository. https://github.com/neilsagarwal/boggart.

[10] British transport police: Cctv. http://www.btp.police.uk/advice_and_information/safety_on_and_near_the_railway/cctv.aspx.

[11] Can 30,000 cameras help solve chicago's crime problem? https://www.nytimes.com/2018/05/26/us/chicago-police-surveillance.html.

[12] City of Auburn Toomer's Corner Webcam 1. https://www.youtube.com/watch?v=wVDtzDwo-1Q.

[13] Computer Vision AI. https://techsee.me/computer-vision/.

[14] Global Sports Analytics Market Size Report, 2021-2028. https://www.grandviewresearch.com/industry-analysis/sports-analytics-market.

[15] Hamptons.com Southampton Village Cam, Hildreth's Home Goods LIVE. https://www.youtube.com/watch?v=9IbruokZzx0.

[16] How to Be Ahead of Your Competition with Data. https://www.hudl.com/blog/how-to-be-ahead-of-your-competition-with-data.

[17] Jackson Hole Wyoming USA Town Square Live Cam. https://www.youtube.com/watch?v=1EiC9bvVGnk.

[18] Lausanne, pont Bessières. https://www.youtube.com/watch?v=TyElel0QjCI.

[19] Live BACKYARD Animal Cam in Ohio! . https://www.youtube.com/watch?v=OIqUka8BOS8.

[20] One traffic framework. Any video source. All traffic tasks. https://datafromsky.com/.

[21] Oxford Martin School Webcam - Broad Street, Oxford. https://www.youtube.com/watch?v=St7aTfoIdYQ.

[22] Paris hospitals to get 1,500 cctv cameras to combat violence against staff. https://bit.ly/2OYiBz2.

[23] Powering the edge with ai in an iot world. https://www.forbes.com/sites/forbestechcouncil/2020/04/06/powering-the-edge-with-ai-in-an-iot-world/.

[24] Resorts Casino Hotel Beach Camera. https://www.youtube.com/watch?v=vVyBOU9Huvo.

[25] SouthHampton Traffic Cam. https://www.youtube.com/watch?v=Z9P_2pCgfBA.

[26] The Hudl Algorithm: Turning Video into Player Tracking Data. https://www.maryecollins.com/hudl-tracking.

[27] Toyota Research Institute accelerates safe automated driving with deep learning. https://www.wired.com/brandlab/2018/08/tri-accelerates-safe-automated-driving-deep-learning-2/.

[28] Unique web-based facial recognition tool enhances security and fights crime. https://www.securityinfowatch.com/access-identity/biometrics/facial-recognition-solutions/article/21261325/unique-webbased-facial-recognition-tool-enhances-security-and-fights-crime.

[29] Using Deep Learning to Find Basketball Highlights. https://www.hudl.com/bits/using-deep-learning-to-find-basketball-highlights.

[30] Venice Italy Live Camera - Grand Canal. https://www.youtube.com/watch?v=P393gTj527k.

[31] Video analytics applications in retail - beyond security. https://www.securityinformed.com/insights/co-2603-ga-co-2214-ga-co-1880-ga.16620.html/.

[32] Video Analytics Market - Growth, Trends, COVID-19 Impact, and Forecasts (2022 - 2027). https://www.mordorintelligence.com/industry-reports/video-analytics-market.

[33] The vision zero initiative. http://www.visionzeroinitiative.com/.

[34] How retail stores can streamline operations with video content analytics. https://www.briefcam.com/resources/blog/how-retail-stores-can-streamline-operations-with-video-content-analytics/, 2020.

[35] Video analytics traffic study creates baseline for change. https://www.govtech.com/analytics/Video-Analytics-Traffic-Study-Creates-Baseline-for-Change.html, 2020.

[36] Ekya: Continuous learning of video analytics models on edge compute servers. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pages 119–135, Renton, WA, Apr. 2022. USENIX Association.

[37] Video analytics market. https://www.fortunebusinessinsights.com/industry-reports/video-analytics-market-101114, 2022.

[38] J. Albericio, A. Delmás, P. Judd, S. Sharify, G. O'Leary, R. Genov, and A. Moshovos. Bit-

pragmatic deep neural network computing. In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-50 '17, page 382–394. Association for Computing Machinery, 2017.

[39] Amazon. Rekognition. https://aws.amazon.com/rekognition/.

[40] Amazon. AWS DeepLens. https://aws.amazon.com/deeplens/, 2019.

[41] G. Ananthanarayanan, Y. Shu, M. Kasap, A. Kewalramani, M. Gada, and V. Bahl. Live video analytics with microsoft rocket for reducing edge compute costs, July 2020.

[42] M. R. Anderson, M. J. Cafarella, G. Ros, and T. F. Wenisch. Physical representation-based predicate optimization for a visual analytics database. In *35th IEEE International Conference on Data Engineering, ICDE 2019, Macao, China, April 8-11, 2019*, pages 1466–1477, 2019.

[43] K. Apicharttrisorn, X. Ran, J. Chen, S. V. Krishnamurthy, and A. K. Roy-Chowdhury. Frugal following: Power thrifty object detection and tracking for mobile augmented reality. In *Proceedings of the 17th Conference on Embedded Networked Sensor Systems*, SenSys '19, page 96–109, New York, NY, USA, 2019. Association for Computing Machinery.

[44] M. Azure. Computer vision api. https://azure.microsoft.com/en-us/services/cognitive-services/computer-vision/, 2021.

[45] M. Azure. Face api. https://azure.microsoft.com/en-us/services/cognitive-services/face/, 2021.

[46] O. Barnich and M. Van Droogenbroeck. Vibe: A universal background subtraction algorithm for video sequences. *IEEE Transactions on Image processing*, 20(6):1709–1724, 2010.

[47] D. Barrett. One surveillance camera for every 11 people in Britain, says CCTV survey. https://www.telegraph.co.uk/technology/10172298/\One-surveillance-camera-for-every-11-people-in-Britain\-says-CCTV-survey.html, 2013.

[48] F. Bastani and S. Madden. Otif: Efficient tracker preprocessing over large video datasets. In *Proceedings of the 2022 International Conference on Management of Data*, SIGMOD '22, 2022.

[49] G. Bertasius, L. Torresani, and J. Shi. Object detection in video with spatiotemporal sampling networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 331–346, 2018.

[50] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft. Simple online and realtime tracking. In *2016 IEEE international conference on image processing (ICIP)*, pages 3464–3468. IEEE, 2016.

[51] D. Blalock, J. J. G. Ortiz, J. Frankle, and J. Guttag. What is the state of neural network pruning? *arXiv preprint arXiv:2003.03033*, 2020.

[52] S. Brutzer, B. Hoferlin, and G. Heidemann. Evaluation of background subtraction techniques for video surveillance. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 1937–1944, Washington, DC, USA, 2011. IEEE Computer Society.

[53] Z. Cai, M. Saberian, and N. Vasconcelos. Learning complexity-aware cascades for deep pedestrian detection. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, pages 3361–3369, Washington, DC, USA, 2015. IEEE Computer Society.

[54] C. Canel, T. Kim, G. Zhou, C. Li, H. Lim, D. G. Andersen, M. Kaminsky, and S. R. Dulloor. Scaling video analytics on constrained edge nodes. In *2nd SysML Conference*, 2019.

[55] F. Cangialosi, N. Agarwal, V. Arun, J. Jiang, S. Narayana, A. Sarwate, and R. Netravali. Privid: Practical, privacy-preserving video analytics queries. In *Proceedings of the 19th USENIX Conference on Networked Systems Design and Implementation*, NSDI'22, Berkeley, CA, USA, 2022. USENIX Association.

[56] Y. Chai. Patchwork: A patch-wise attention network for efficient object detection and segmentation in video streams. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3415–3424, 2019.

[57] K. Chen, J. Wang, S. Yang, X. Zhang, Y. Xiong, C. C. Loy, and D. Lin. Optimizing video object detection via a scale-time lattice. In *CVPR*, 2018.

[58] T. Y.-H. Chen, L. Ravindranath, S. Deng, P. Bahl, and H. Balakrishnan. Glimpse: Continuous, real-time object recognition on mobile devices. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, pages 155–168, 2015.

[59] Y. Chen, Y. Cao, H. Hu, and L. Wang. Memory enhanced global-local aggregation for video object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10337–10346, 2020.

[60] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

[61] S. R. E. Datondji, Y. Dupuis, P. Subirats, and P. Vasseur. A survey of vision-based traffic monitoring of road intersections. *Trans. Intell. Transport. Sys.*, 17(10):2681–2698, Oct. 2016.

[62] J. Deng, Y. Pan, T. Yao, W. Zhou, H. Li, and T. Mei. Relation distillation networks for video object detection. In *Proceedings of the IEEE/CVF International*

Conference on Computer Vision, pages 7023–7032, 2019.

[63] U. Drolia, K. Guo, J. Tan, R. Gandhi, and P. Narasimhan. Cachier: Edge-caching for recognition applications. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 276–286, 2017.

[64] K. Du, A. Pervaiz, X. Yuan, A. Chowdhery, Q. Zhang, H. Hoffmann, and J. Jiang. Server-driven video streaming for deep learning inference. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '20, page 557–570, New York, NY, USA, 2020. Association for Computing Machinery.

[65] A. Elqursh and A. Elgammal. Online moving camera background subtraction. In A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, editors, *Computer Vision – ECCV 2012*, pages 228–241, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[66] J. Emmons, S. Fouladi, G. Ananthanarayanan, S. Venkataraman, S. Savarese, and K. Winstein. Cracking open the dnn black-box: Video analytics with dnns across the camera-cloud boundary. In *Proceedings of the 2019 Workshop on Hot Topics in Video Analytics and Intelligent Edges*, HotEdgeVideo'19, pages 27–32, New York, NY, USA, 2019. Association for Computing Machinery.

[67] M. Everingham, L. Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vision*, 88(2):303–338, June 2010.

[68] J. Fowers, K. Ovtcharov, M. Papamichael, T. Massengill, M. Liu, D. Lo, S. Alkalay, M. Haselman, L. Adams, M. Ghandi, S. Heil, P. Patel, A. Sapek, G. Weisz, L. Woods, S. Lanka, S. K. Reinhardt, A. M. Caulfield, E. S. Chung, and D. Burger. A configurable cloud-scale dnn processor for real-time ai. In *Proceedings of the 45th Annual International Symposium on Computer Architecture*, ISCA '18, page 1–14. IEEE Press, 2018.

[69] I. Ghodgaonkar, S. Chakraborty, V. Banna, S. Allcroft, M. Metwaly, F. Bordwell, K. Kimura, X. Zhao, A. Goel, C. Tung, et al. Analyzing worldwide social distancing through large-scale computer vision. *arXiv preprint arXiv:2008.12363*, 2020.

[70] Google. Cloud vision api. https://cloud.google.com/vision, 2021.

[71] C. Grana, D. Borghesani, and R. Cucchiara. Optimized block-based connected components labeling with decision trees. *IEEE Transactions on Image Processing*, 19(6):1596–1609, 2010.

[72] C. Guo, B. Fan, J. Gu, Q. Zhang, S. Xiang, V. Prinet, and C. Pan. Progressive sparse local attention for video object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3909–3918, 2019.

[73] P. Guo, B. Hu, R. Li, and W. Hu. Foggycache: Cross-device approximate computation reuse. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, MobiCom '18, page 19–34, New York, NY, USA, 2018. Association for Computing Machinery.

[74] P. Guo and W. Hu. *Potluck: Cross-Application Approximate Deduplication for Computation-Intensive Mobile Applications*, page 271–284. Association for Computing Machinery, New York, NY, USA, 2018.

[75] S. Han, H. Shen, M. Philipose, S. Agarwal, A. Wolman, and A. Krishnamurthy. MCDNN: An approximation-based execution framework for deep stream processing under resource constraints. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '16, pages 123–136, New York, NY, USA, 2016. ACM.

[76] F. He, N. Gao, Q. Li, S. Du, X. Zhao, and K. Huang. Temporal context enhanced feature aggregation for video object detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 10941–10948, 2020.

[77] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.

[78] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[79] D. Hoiem, Y. Chodpathumwan, and Q. Dai. Diagnosing error in object detectors. In *European conference on computer vision*, pages 340–353. Springer, 2012.

[80] K. Hsieh, G. Ananthanarayanan, P. Bodik, S. Venkataraman, P. Bahl, M. Philipose, P. B. Gibbons, and O. Mutlu. Focus: Querying large video datasets with low latency and low cost. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 269–286, Carlsbad, CA, 2018. USENIX Association.

[81] B. Hu, P. Guo, and W. Hu. Video-zilla: An indexing layer for scalable live video analytics. In *Proceedings of the 2022 International Conference on Management of Data*, SIGMOD '22, 2022.

[82] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy. Speed/accuracy trade-offs for modern convolutional object detectors. *CoRR*, abs/1611.10012, 2016.

[83] IBM. Maximo remote monitoring. https://www.ibm.com/products/maximo/remote-monitoring, 2021.

[84] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang,

A. Howard, H. Adam, and D. Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2704–2713, 2018.

[85] P. Jain, X. Mo, A. Jain, H. Subbaraj, R. S. Durrani, A. Tumanov, J. Gonzalez, and I. Stoica. Dynamic space-time scheduling for gpu inference. *arXiv preprint arXiv:1901.00041*, 2018.

[86] S. Jain, X. Zhang, Y. Zhou, G. Ananthanarayanan, J. Jiang, Y. Shu, V. Bahl, and J. Gonzalez. Spatula: Efficient cross-camera video analytics on large camera networks. In *ACM/IEEE Symposium on Edge Computing (SEC 2020)*, November 2020.

[87] J. Jiang, G. Ananthanarayanan, P. Bodik, S. Sen, and I. Stoica. Chameleon: Scalable adaptation of video analytics. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '18, pages 253–266, New York, NY, USA, 2018. ACM.

[88] J. Jodoin, G. Bilodeau, and N. Saunier. Urban tracker: Multiple object tracking in urban mixed traffic. In *IEEE Winter Conference on Applications of Computer Vision*, pages 885–892, 2014.

[89] J. Jodoin, G. Bilodeau, and N. Saunier. Tracking all road users at multimodal urban traffic intersections. *IEEE Transactions on Intelligent Transportation Systems*, 17(11):3241–3251, 2016.

[90] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, R. Boyle, P.-l. Cantin, C. Chao, C. Clark, J. Coriell, M. Daley, M. Dau, J. Dean, B. Gelb, T. V. Ghaemmaghami, R. Gottipati, W. Gulland, R. Hagmann, C. R. Ho, D. Hogberg, J. Hu, R. Hundt, D. Hurt, J. Ibarz, A. Jaffey, A. Jaworski, A. Kaplan, H. Khaitan, D. Killebrew, A. Koch, N. Kumar, S. Lacy, J. Laudon, J. Law, D. Le, C. Leary, Z. Liu, K. Lucke, A. Lundin, G. MacKean, A. Maggiore, M. Mahony, K. Miller, R. Nagarajan, R. Narayanaswami, R. Ni, K. Nix, T. Norrie, M. Omernick, N. Penukonda, A. Phelps, J. Ross, M. Ross, A. Salek, E. Samadiani, C. Severn, G. Sizikov, M. Snelham, J. Souter, D. Steinberg, A. Swing, M. Tan, G. Thorson, B. Tian, H. Toma, E. Tuttle, V. Vasudevan, R. Walter, W. Wang, E. Wilcox, and D. H. Yoon. In-datacenter performance analysis of a tensor processing unit. *SIGARCH Comput. Archit. News*, 45(2):1–12, June 2017.

[91] C. Ju, Z. Wang, C. Long, X. Zhang, G. Cong, and D. E. Chang. Interaction-aware kalman neural networks for trajectory prediction. *CoRR*, abs/1902.10928, 2019.

[92] D. Kang, P. Bailis, and M. Zaharia. Blazeit: Fast exploratory video queries using neural networks. *CoRR*, abs/1805.01046, 2018.

[93] D. Kang, P. Bailis, and M. Zaharia. Blazeit: Optimizing declarative aggregation and limit queries for neural network-based video analytics. *Proc. VLDB Endow.*, 13(4):533–546, Dec. 2019.

[94] D. Kang, J. Emmons, F. Abuzaid, P. Bailis, and M. Zaharia. Noscope: Optimizing neural network queries over video at scale. *Proc. VLDB Endow.*, 10(11):1586–1597, Aug. 2017.

[95] D. Kang, J. Guibas, P. Bailis, T. Hashimoto, and M. Zaharia. Task-agnostic indexes for deep learning-based queries over unstructured data, 2020.

[96] D. Kang, A. Mathur, T. Veeramacheneni, P. Bailis, and M. Zaharia. Jointly optimizing preprocessing and inference for dnn-based visual analytics. *Proc. VLDB Endow.*, 14(2):87–100, Oct. 2020.

[97] K. Kang, H. Li, J. Yan, X. Zeng, B. Yang, T. Xiao, C. Zhang, Z. Wang, R. Wang, X. Wang, and W. Ouyang. T-CNN: Tubelets With Convolutional Neural Networks for Object Detection From Videos. *IEEE Trans. Cir. and Sys. for Video Technol.*, 28(10):2896–2907, Oct. 2018.

[98] K. Kang, W. Ouyang, H. Li, and X. Wang. Object detection from video tubelets with convolutional neural networks. In *CVPR*, 2016.

[99] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, May 2017.

[100] B. Kueng, E. Mueggler, G. Gallego, and D. Scaramuzza. Low-latency visual odometry using event-based feature tracks. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 16–23, Oct 2016.

[101] B. Laugraud, S. Piérard, and M. Van Droogenbroeck. Labgen: A method based on motion detection for generating the background of a scene. *Pattern Recognition Letters*, 96:12–21, 2017.

[102] J. Le. Part 1: An overview of dataops for computer vision. https://www.superb-ai.com/blog/part-1-an-overview-of-dataops-for-computer-vision, 2021.

[103] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[104] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua. A convolutional neural network cascade for face detection. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5325–5334, June 2015.

[105] Y. Li, A. Padmanabhan, P. Zhao, Y. Wang, G. H. Xu, and R. Netravali. Reducto: On-Camera Filtering for Resource-Efficient Real-Time Video Analytics. SIGCOMM '20, page 359–376, New York, NY, USA, 2020. Association for Computing Machinery.

[106] T. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–944, July 2017.

[107] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen. Deep learning for generic object detection: A survey, 2019.

[108] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2736–2744, 2017.

[109] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, Los Alamitos, CA, USA, jun 2015. IEEE Computer Society.

[110] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, Nov. 2004.

[111] Y. Lu, A. Chowdhery, and S. Kandula. Optasia: A relational platform for efficient large-scale video analytics. In *Proceedings of the Seventh ACM Symposium on Cloud Computing*, SoCC '16, pages 57–70, New York, NY, USA, 2016. ACM.

[112] H. Mao, T. Kong, and W. J. Dally. Catdet: Cascaded tracked detector for efficient object detection from video. *arXiv preprint arXiv:1810.00434*, 2018.

[113] A. Mhalla, T. Chateau, H. Maamatou, S. Gazzah, and N. E. B. Amara. Smc faster r-cnn: Toward a scene-specialized multi-object detector. *Computer Vision and Image Understanding*, 164:3–15, 2017.

[114] A. Moschitti. Updating neural networks to recognize new categories, with minimal retraining. https://www.amazon.science/blog/updating-neural-networks-to-recognize-new-categories-with-minimal-retraining, 2019.

[115] OpenCV. Morphological Transformations. https://docs.opencv.org/master/d9/d61/tutorial_py_morphological_ops.html, 2020.

[116] A. Padmanabhan, N. Agarwal, A. Iyer, G. Ananthanarayanan, Y. Shu, N. Karianakis, G. H. Xu, and R. Netravali. Gemel: Model merging for memory-efficient, real-time video analytics at the edge, 2022.

[117] S. Park, J. Park, K. Bong, D. Shin, J. Lee, S. Choi, and H. Yoo. An energy-efficient and scalable deep learning/inference processor with tetra-parallel mimd architecture for big data applications. *IEEE Transactions on Biomedical Circuits and Systems*, 9(6):838–848, 2015.

[118] R. Poddar, G. Ananthanarayanan, S. Setty, S. Volos, and R. A. Popa. Visor: Privacy-preserving video analytics as a cloud service. In S. Capkun and F. Roesner, editors, *29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020*, pages 1039–1056. USENIX Association, 2020.

[119] A. Poms, W. Crichton, P. Hanrahan, and K. Fatahalian. Scanner: Efficient video analysis at scale. *ACM Trans. Graph.*, 37(4), July 2018.

[120] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[121] W. Ren, S. Singh, M. Singh, and Y. S. Zhu. State-of-the-art on spatio-temporal information-based video retrieval. *Pattern Recogn.*, 42(2):267–282, Feb. 2009.

[122] A. Rizzoli. 7 Game-Changing AI Applications in the Sports Industry. https://www.v7labs.com/blog/ai-in-sports, 2022.

[123] H. Shen, L. Chen, Y. Jin, L. Zhao, B. Kong, M. Philipose, A. Krishnamurthy, and R. Sundaram. Nexus: A gpu cluster engine for accelerating dnn-based video analysis. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, SOSP '19, pages 322–337, New York, NY, USA, 2019. Association for Computing Machinery.

[124] C. Stauffer and W. E. L. Grimson. Adaptive background mixture models for real-time tracking. In *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, volume 2, pages 246–252 Vol. 2, 1999.

[125] Y. Sun, X. Wang, and X. Tang. Deep convolutional network cascade for facial point detection. In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '13, pages 3476–3483, Washington, DC, USA, 2013. IEEE Computer Society.

[126] Y. Ukidave, X. Li, and D. Kaeli. Mystic: Predictive scheduling for gpu based cloud servers using machine learning. In *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 353–362. IEEE, 2016.

[127] P. D. Z. Varcheie, M. Sills-Lavoie, and G.-A. Bilodeau. A multiscale region-based motion detection and background subtraction algorithm. *Sensors*, 10(2):1041–1061, 2010.

[128] A. Viswanath, R. K. Behera, V. Senthamilarasu, and K. Kutty. Background modelling from a moving camera. volume 58, pages 289–296, 2015. Second International Symposium on Computer Vision and the Internet (VisionNet'15).

[129] J. Wang, Z. Feng, Z. Chen, S. George, M. Bala, P. Pillai, S.-W. Yang, and M. Satyanarayanan. Bandwidth-efficient live video analytics for drones via edge computing. pages 159–173, 10 2018.

[130] S. Wang, H. Lu, and Z. Deng. Fast object detection in compressed video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages

7104–7113, 2019.

[131] S. Wang, Y. Zhou, J. Yan, and Z. Deng. Fully motion-aware network for video object detection. In *Proceedings of the European conference on computer vision (ECCV)*, pages 542–557, 2018.

[132] X. Wang, T. Xiao, Y. Jiang, S. Shao, J. Sun, and C. Shen. Repulsion loss: Detecting pedestrians in a crowd. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7774–7783, 2018.

[133] Y. Wang, W. Wang, J. Zhang, J. Jiang, and K. Chen. Bridging the edge-cloud barrier for real-time advanced vision analytics. In *11th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 19)*, Renton, WA, July 2019. USENIX Association.

[134] N. Wojke, A. Bewley, and D. Paulus. Simple online and realtime tracking with a deep association metric, 2017.

[135] S. Xie, W. Zhang, W. Ying, and K. Zakim. Fast detecting moving objects in moving background using orb feature matching. In *2013 Fourth International Conference on Intelligent Control and Information Processing (ICICIP)*, pages 304–309, 2013.

[136] M. Xu, M. Zhu, Y. Liu, F. X. Lin, and X. Liu. Deepcache: Principled cache for mobile deep vision. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, MobiCom '18, page 129–144, New York, NY, USA, 2018. Association for Computing Machinery.

[137] T. Xu, L. M. Botelho, and F. X. Lin. Vstore: A data store for analytics on large videos. In *Proceedings of the Fourteenth EuroSys Conference 2019*, EuroSys '19, pages 16:1–16:17, New York, NY, USA, 2019. ACM.

[138] Q. Xue, X. Li, J. Zhao, and W. Zhang. Deep kalman filter: A refinement module for the rollout trajectory prediction methods. *CoRR*, abs/2102.10859, 2021.

[139] J. Yuan, H. Wang, L. Xiao, W. Zheng, J. Li, F. Lin, and B. Zhang. A formal study of shot boundary detection. *IEEE Trans. Cir. and Sys. for Video Technol.*, 17(2):168–186, Feb. 2007.

[140] H. Zhang, G. Ananthanarayanan, P. Bodik, M. Philipose, P. Bahl, and M. J. Freedman. Live video analytics at scale with approximation and delay-tolerance. In *Proceedings of the 14th USENIX Conference on Networked Systems Design and Implementation*, NSDI'17, pages 377–392, Berkeley, CA, USA, 2017. USENIX Association.

[141] T. Zhang, A. Chowdhery, P. Bahl, K. Jamieson, and S. Banerjee. The design and implementation of a wireless video surveillance system. pages 426–438, 09 2015.

[142] D. Zhou, L. Wang, X. Cai, and Y. Liu. Detection of moving targets with a moving camera. In *2009 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 677–681, 2009.

[143] X. Zhou, X. Zhou, L. Chen, and A. Bouguettaya. Efficient subsequence matching over large video databases. *The VLDB Journal*, 21(4):489–508, Aug. 2012.

[144] C. Zhu, S. Han, H. Mao, and W. J. Dally. Trained ternary quantization. *arXiv preprint arXiv:1612.01064*, 2016.

[145] X. Zhu, J. Dai, L. Yuan, and Y. Wei. Towards high performance video object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7210–7218, 2018.

[146] X. Zhu, Y. Wang, J. Dai, L. Yuan, and Y. Wei. Flow-guided feature aggregation for video object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 408–417, 2017.

[147] X. Zhu, Y. Xiong, J. Dai, L. Yuan, and Y. Wei. Deep feature flow for video recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2349–2358, 2017.

[148] Y. Zhu, A. Samajdar, M. Mattina, and P. Whatmough. Euphrates: Algorithm-soc co-design for low-power mobile continuous vision. In *Proceedings of the 45th Annual International Symposium on Computer Architecture*, ISCA '18, page 547–560. IEEE Press, 2018.

[149] Z. Zou, Z. Shi, Y. Guo, and J. Ye. Object detection in 20 years: A survey, 2019.