VOCAL: Video Organization and Interactive Compositional AnaLytics

Vision Paper

Maureen Daum[*]
mdaum@cs.washington.edu
University of Washington

Enhao Zhang[*]
enhaoz@cs.washington.edu
University of Washington

Dong He
donghe@cs.washington.edu
University of Washington

Magdalena Balazinska
magda@cs.washington.edu
University of Washington

Brandon Haynes
brandon.haynes@microsoft.com
Microsoft, Gray Systems Lab

Ranjay Krishna
ranjaykrishna@cs.washington.edu
University of Washington

Apryle Craig
apryle@uw.edu
University of Washington

Aaron Wirsing
wirsinga@uw.edu
University of Washington

ABSTRACT

Current video database management systems (VDBMSs) fail to support the growing number of video datasets in diverse domains because these systems assume clean data and rely on pretrained models to detect known objects or actions. Existing systems also lack good support for compositional queries that seek events consisting of multiple objects with complex spatial and temporal relationships. In this paper, we propose VOCAL, a vision of a VDBMS that supports efficient data cleaning, exploration and organization, and compositional queries, even when no pretrained model exists to extract semantic content. These techniques utilize optimizations to minimize the manual effort required of users.

1 INTRODUCTION

In many application domains—ranging from scientific research to urban planning—the use of large video datasets has become common in great part because camera quality continues to improve and storage remains affordable. Video datasets capture content from diverse domains and support widely varying applications, such as using cameras mounted on vehicles to understand how neighborhoods evolved through the pandemic [39], using animal-borne cameras to investigate animal (e.g., shark) habitat preferences [28], or using citywide camera networks to measure the effect of traffic interventions on crowded city streets [55]. In response to the proliferation of video datasets and applications, recent work has pushed the state of the art in the field of video database management systems (VDBMSs) [1, 12, 23, 31, 37]. These systems, however, (i) expect clean video data, (ii) require pretrained machine learning models to detect objects or actions, and (iii) have limited or no support for compositional queries about multiple objects interacting over time.

[*]Both authors contributed equally to the paper

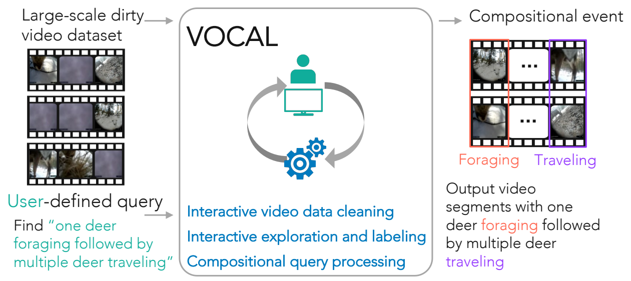

Large-scale dirty video dataset
VOCAL
Compositional event
Foraging
Traveling
User-defined query
Find "one deer foraging followed by multiple deer traveling"
Interactive video data cleaning
Interactive exploration and labeling
Compositional query processing
Output video segments with one deer foraging followed by multiple deer traveling

Figure 1: We introduce VOCAL, an end-to-end video analytics system that facilitates cleaning, exploration, and compositional query processing over large-scale video datasets, even for domains where no pretrained models exist.

In contrast, most real world collection efforts violate all three assumptions. They result in videos with dirty data. They often lack domain-specific models to support the target application. Users would like to ask complex, compositional queries on their data. As a motivating example, we are working with a dataset of videos captured by cameras attached to wild deer (*Odocoileus spp.*) [16]. The videos are to be used to provide insight into deer activity patterns: How much time do deer spend foraging versus lying down? How do environmental conditions (e.g., slope and depth of snow) affect deer activity? The collected data is dirty: camera lenses are often obscured by snow for a large fraction of time (up to 75%), requiring that scientists first manually clean each video to find usable fragments. Because egocentric deer video is an uncommon domain, there are no pretrained machine learning models available to detect deer activity directly. Scientists thus need to annotate their videos manually. This is tedious, so scientists have only annotated a small fraction of the data. Finally, scientists would like to understand complex deer behavior that goes beyond simple activity detection.

In this paper, we propose VOCAL (Video Organization and Interactive Compositional AnaLytics), a vision of a VDBMS that supports user-defined applications and compositional queries for *any* dataset, even when the videos are dirty or lack domain-specific pretrained models (Figure 1). We contribute initial techniques towards developing VOCAL to (i) filter dirty data from videos (Section 2), (ii)

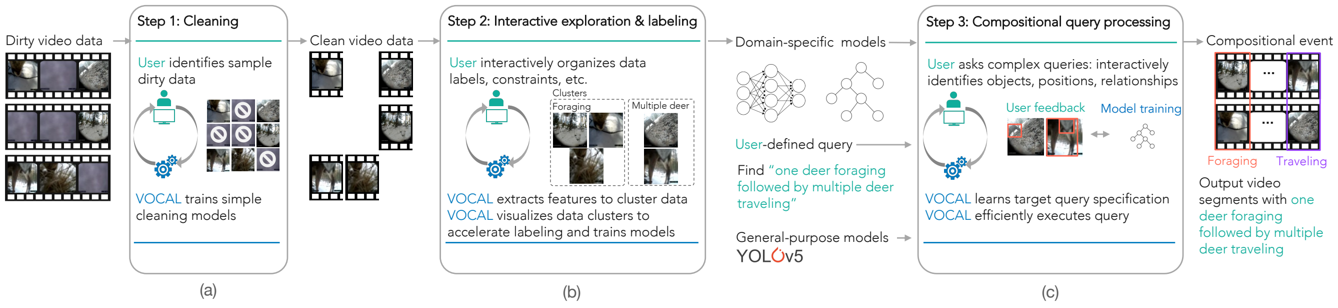

Dirty video data
Step 1: Cleaning
User identifies sample dirty data
VOCAL trains simple cleaning models
(a)
Clean video data
Step 2: Interactive exploration & labeling
User interactively organizes data labels, constraints, etc.
Clusters
Foraging
Multiple deer
VOCAL extracts features to cluster data
VOCAL visualizes data clusters to accelerate labeling and trains models
(b)
Domain-specific models
User-defined query
Find "one deer foraging followed by multiple deer traveling"
General-purpose models
YOLOv5
Step 3: Compositional query processing
User asks complex queries: interactively identifies objects, positions, relationships
User feedback
Model training
VOCAL learns target query specification
VOCAL efficiently executes query
(c)
Compositional event
...
Foraging
Traveling
Output video segments with one deer foraging followed by multiple deer traveling

Figure 2: VOCAL consists of three steps. (a) First, VOCAL supports data cleaning to remove dirty video data. (b) Second, VOCAL enables exploring and labeling the clean video data to train domain-specific models. (c) Third, VOCAL utilizes these domain-specific models to perform compositional query processing over the video data.

support model development through exploration and organization of videos using an interactive clustering algorithm (Section 3), and (iii) evaluate compositional queries by iteratively learning and refining the event specification (Section 4). These techniques and their interactions are illustrated in Figure 2.

Prior to analyzing videos, users must filter dirty video segments (Figure 2a). Dirty video data may be generated in many ways. For example, the camera may malfunction and record when the user did not intend (e.g., at night), environmental forces may misalign the camera’s field of view, or the camera lens might be obscured. Without support for data cleaning from existing VDBMSs, filtering dirty data is a tedious manual process. The user likely does not know all the causes of data collection failure, especially when there are too many videos to watch in their entirety. We observe, however, that the types of video segments we are trying to filter out are identifiable by looking at a few frames (e.g., segments where the camera lens is completely covered by snow). Therefore, we propose learning simple classifiers using basic color and motion features, which are cheap to extract. These models require minimal user annotations and are fast to train. To do this, as users iteratively mark dirty video segments, VOCAL’s cleaning subsystem trains a classifier that is used to automatically remove similar segments. The output of this step is a clean dataset.

Once the dataset is cleaned, VOCAL’s data exploration subsystem (Figure 2b) supports investigative analyses by allowing users to explore, organize, and label video segments with information, such as entities, events, or environmental conditions, to build domain-specific models. Today, users must manually annotate large datasets to build such models if they do not exist yet. We seek to reduce this manual effort. The first challenge is feature extraction. Simple color or motion features may be too limited to distinguish events of interest. Features from existing pretrained models from other domains may not be applicable. For example, egocentric cameras are mobile, complicating the identification of important visual features. The second challenge is an evolving label set. When initially working with a new dataset, users may be unsure of what events occur and what they look like from the perspective of the camera, thus updating their desired set of labels while exploring the data.

To address these challenges, we propose an active learning approach that reduces manual efforts by prompting the user to only annotate the most useful video segments. Behind our active learning approach lies a clustering technique to partition the dataset into clusters of video segments with visually or semantically similar content. VOCAL shows the most informative clusters to users, who iteratively merge and split them to create useful labels. The output of this step is a set of clusters that can be used to train domain-specific models to automatically identify video events. These models are used as input to the following step.

Finally, VOCAL supports complex inquiry by enabling *compositional query processing* (Figure 2c). For instance, scientists may query for video where a deer was grazing, then sighted a predator, and finally began running. Query specification is the primary challenge of compositional queries because users must express queries in a manner that encompasses an event’s high variance (e.g., a deer may graze for one minute or one hour on grass or snowy terrain, then sight a predator). Models today can detect only a limited set of activities in videos [47] and are expensive to evaluate. Thus, most existing systems supporting spatiotemporal compositional queries [23, 37, 54] still rely on models that process single frames, detect all objects before query execution, and require precise query specifications (we discuss related work in more detail in Section 5).

To address these challenges, we introduce an iterative query refinement approach that automatically learns the event specification from user feedback. VOCAL iteratively presents frames that the user annotates with objects, positions, and relationships. VOCAL trains simple classifiers to learn the event specification using spatial configurations extracted from the feedback because, as discussed above, simple color and motion features fail to capture semantic specifications from the annotations. To minimize the user’s annotation efforts, VOCAL prioritizes frames using temporal sampling based on event duration and predictions from simple classifiers. We also explore techniques to accelerate compositional query execution.

While we focus on the deer video dataset, these three steps generalize to other domains. The second application we explore in this paper is that of a city planner who seeks to utilize traffic cameras to investigate whether allowing cars to turn right on red lights creates hazards for pedestrians. The city planner must filter out video data where the camera is obscured, for example, by heavy rain. They must explore the video to learn how vehicles and pedestrians move through the intersection. They can finally query for their events of interest.

While the initial techniques presented in this paper are evaluated on two datasets that correspond to our two running examples, our

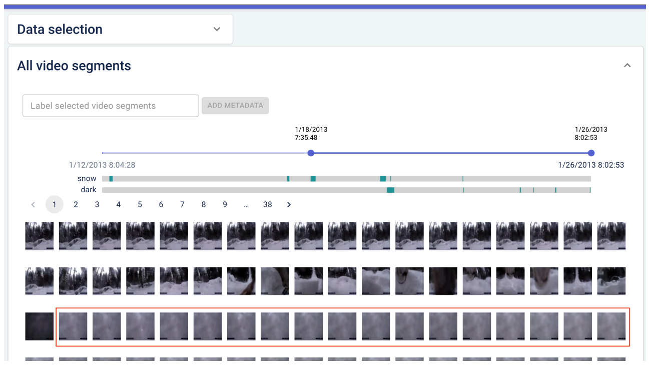


Figure 3: Exploration tool interface. The segments inside of the red box would be identified as dirty data.

ultimate goal is to develop techniques that are applicable to any video dataset where there are clearly identifiable entities, events, or environmental conditions.

2 DATA CLEANING

In many video datasets, some amount of data is corrupted or has captured something other than the input of interest. Data cleaning is therefore necessary to remove this dirty data before users can proceed with their analysis, as illustrated in Figure 2a.

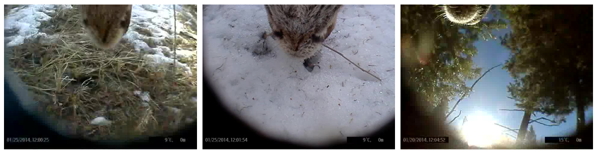
Example: In the deer video dataset, between 0.4% and 75% of the data for each deer, which corresponds to 0.2–35 hours of video data, was either obscured by snow or unintentionally captured at night, and therefore was not usable.

Requirements: VOCAL should support users in effectively removing dirty data. It should automatically identify clean and dirty data given a small number of user-provided annotations. As the user provides more annotations, VOCAL's ability to detect dirty data should improve.

Challenges: The first challenge (**C1**) is to support users in efficiently finding examples of dirty data in large video datasets. The second challenge (**C2**) is quickly updating models in response to additional data.

Approach: Our approach to address challenge **C1** is to first show users a high-level summary of the data. To do this, VOCAL first splits long videos into short segments. For the deer videos, the data already comes in 10-second segments. As illustrated in Figure 3, VOCAL initially shows only the first frame of each segment and the user can scroll rapidly through the collection of segments. For many segments, the first frame suffices to identify dirty data segments. If the user is not sure of their label because a segment is directly followed by something interesting, they can hover over the frame to play the entire segment before labeling it.

VOCAL then builds a classifier to isolate the unannotated dirty data. To address challenge **C2** and ensure rapid data cleaning with little manual effort, our current prototype uses a support vector machine (SVM) classifier, which is fast to train and requires a small number of labels compared to more complex models such as a deep neural network (DNN). The model further uses basic features such as color and motion, which are fast to extract. This approach works because dirty data is typically very different from good data.



(a) Foraging **(b) Foraging (snowy)** **(c) Bedded**

Figure 4: Sample frames for activities from the deer dataset.

Preliminary results: We train twelve SVM classifiers, one for each individual deer, to detect dirty video segments. We train the SVM on the following features (extracted using OpenCV [6]) in each 10-second video segment: average luminance extracted from the median frame, the amplitude of the 2-dimensional discrete Fourier transform, the changes in average gray-level values and intensity distribution across frames using a 3×3 frame partitioning as in [46], and changes along the horizontal and vertical axes across frames computed with dense optical flow. These features capture basic information about the visual content and motion. For each deer, we order the video segments by time and use the last 25% as the test set. We pick 5% of the remaining video segments as the training set using stratified sampling to ensure it contains both clear and obstructed video segments. These classifiers achieve a median accuracy of 98% and a median F1 score of 98%.

While other datasets may have a smaller amount of dirty data or data that is less obviously dirty, we expect to be able to extend this simple classifier technique to detect dirty data in videos where the dirty data is visually distinct from clean data.

3 INTERACTIVE EXPLORATION & LABELING

Following data cleaning and as illustrated in Figure 2b, the next requirement is the ability for users to annotate collected videos with the interesting content that was captured with the goal to train domain-specific models to automatically detect that content. The baseline annotation process that we seek to improve upon consists of users manually sampling video segments then having workers manually annotate each segment. This process is tedious and difficult to repeat due to inter- and intra-worker biases.

To address these challenges, VOCAL employs an interactive data exploration and organization process. Our approach clusters video segments into groups based on visual similarity and supports users in labeling and refining those groups, which then serve as labeled data for training domain-specific models. The output of this step is a set of labeled clusters and trained models that automatically classify video segments following the user-defined labels.

Example: In our running example, scientists seek to identify *activities* that the deer engage in (e.g., foraging, lying down, walking, being alert) and the *environmental conditions* associated with each activity (e.g., terrain characteristics like slope, snow depth). Scientists also need to capture the *duration* of each activity. Activities can have arbitrary lengths, be hierarchically composed, and be overlapping.

Requirements: VOCAL must support users in labeling their data in a way that enables training accurate models for the domain at hand. VOCAL should iteratively incorporate user feedback to refine how data is labeled and the quality of the learned models. Updates in response to user feedback should be fast to support an interactive user experience. The system can neither assume that the dataset is fully annotated nor that the full set of labels is known or fixed. Finally, it is important to minimize the amount of user actions required to derive good labels and models. Therefore, VOCAL should guide users to annotate the segments that are most informative for the target downstream model.

Challenges: Automatically detecting entities, events, and conditions in videos from a new domain poses three challenges. The first challenge (**C1**) is to develop automatic ways of extracting and learning features. Many video datasets, especially scientific ones, capture unique domains that are different from the datasets used to train existing models. For example, the deer dataset contains egocentric videos from the perspective of a deer as shown in Figure 4. This is quite distinct from commonly used models in the consumer or urban setting, such as the Kinetics dataset [7], which consists of YouTube videos of human activities. When the data comes from a unique domain, VOCAL cannot use models pretrained on existing data to extract semantic information. The second challenge (**C2**) is that users may not know ahead of time all the activities that the dataset contains, nor what those activities look like. For example, it is unclear what a foraging deer looks like in an egocentric video prior to watching the video, or if the video will contain examples of the deer running away from a predator. The third challenge (**C3**) is that labeling the video dataset is tedious, especially for activities that occur rarely and at unpredictable times. As a result, obtaining large numbers of labels in order to train a domain-specific model through transfer learning or from scratch is a difficult task.

3.1 User Interactions and Active Learning

To address challenges **C2** and **C3**, we are developing an interactive exploration and organization technique that minimizes annotations by allowing users to work with clusters of video segments instead of individual segments.

We assume we can extract features from video segments (discussed in the next subsection). Then, using these features, VOCAL clusters all of the segments and allows the user to split (manually or automatically), merge, or label each cluster. VOCAL then incorporates this feedback using two phases to decide which clusters to show to the user in the next round of feedback. Picking the best set of clusters to present to the user is important because it determines which clustering errors the user notices, and therefore which corrections they make. In the first phase, VOCAL updates the set of clusters and labeled data. If the user manually split a cluster, the system groups the segments as directed by the user. If the user elects to automatically split a cluster, VOCAL applies a clustering algorithm to just the points in that cluster to break it into smaller clusters, each containing more similar video segments. If the user merges or labels clusters, VOCAL propagates any labels to all other members of these clusters. VOCAL additionally utilizes the temporal relationship between video segments to automatically propagate labels to temporally adjacent segments because nearby segments are likely to correspond to the same activity and conditions. Ideally, by increasing the total number of labels, despite some of the automatically inferred ones being noisy, we can more quickly converge on a useful dataset for training domain-specific models.

In the second phase, VOCAL uses active learning to rank the clusters according to their potential to improve the model being trained using an uncertainty metric [45]. It picks the top-ranked clusters to show to the user in the next round. Rather than ranking all video segments, VOCAL accelerates this step by only considering a single representative from each cluster (the cluster medoid). VOCAL further reduces the set of candidate segments by not considering points that are temporally adjacent to points already constrained by the user, again because events and activities are likely to extend to nearby video segments. Once VOCAL prunes the set of segments the active learning strategy should consider, it incorporates class information if it is learning a multi-class classifier. This is important because class imbalance can hurt the performance of active learning algorithms [18], but we expect videos to contain imbalanced events and that rare events will be of interest to users. In multi-class settings, there are two possible approaches VOCAL can take: (1) sampling uncertain segments along each class decision boundary as in [18], or (2) sampling first along the easiest class decision boundary to build a high-quality binary classifier, then later focusing user effort on distinguishing the more difficult decision boundaries. We are exploring under which conditions each of these strategies performs better so that VOCAL can switch between them automatically. Finally, once VOCAL picks a set of clusters to show to the user, VOCAL further samples a small number of video segments from each to show in the user interface because clusters may be too large to show in their entirety. Our initial approach is to show the cluster medoid, a few segments that are close to the medoid, and a few segments near the boundary that may be misclassified, as we showed this approach to be useful for model debugging in prior work [40].

As discussed, VOCAL operates over clusters rather than individual data points. This has two main benefits. First, it acts as a form of representative sampling for active learning [45], which is a technique that prefers to query points that are both uncertain and similar to other points in the dataset (rather than querying highly uncertain points that are outliers). When clusters contain similar points, feedback users apply to a cluster representative can be propagated to other members of the cluster. This is more efficient than users labeling individual points. Considering just cluster representatives also reduces the total set of segments that have to be considered for each round of active learning, and therefore supports interactive updates. Second, users may want to go beyond classification and add relationships (e.g., subclasses) to labels. Clusters are a natural way to operate over data that will be organized into a taxonomy (e.g., using hierarchical clustering).

Before users can engage in the iterative clustering process described above, VOCAL first supports cold starting the exploration process when there are no labels yet. Our prototype organizes the video segments in temporal order and lets users browse the data and make annotations as they start to understand the video content. We also enable filtering based on previously defined labels and displaying the distribution of previously defined labels (e.g., while users may not currently be viewing any segments showing a traveling deer, they can see that such an activity occurs in future segments). Details of our prototype user interface are shown in Figure 5. Maintaining interactive response times in the interface in response to user actions (e.g., filtering or navigation) is difficult because of the amount of data as well as the inherently high cost of transferring and rendering videos. We are developing a variety

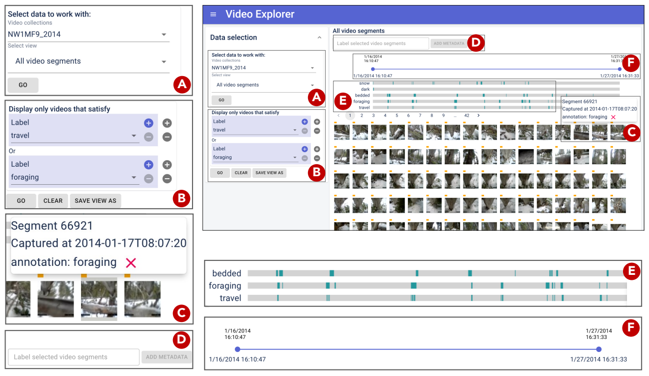
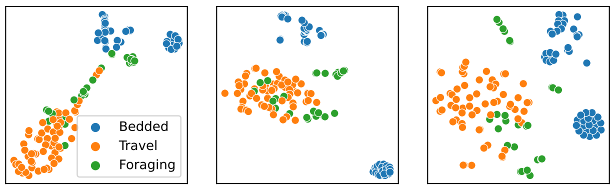


Figure 5: The initial data exploration interface (before clustering). (A) Options to select a deer's data. (B) Options to filter segments the user has manually annotated (example shows selection of travel or foraging segments). (C) Segment details displayed after hovering over a segment. (D) A text field to add annotations to selected segments. (E) Timeline elements showing the distribution of annotations. (F) A slider control to filter video segments by time.



(a) Basic (b) Pretrained (c) Self-supervised

Figure 6: t-SNE visualization of video segments for different feature types. All feature sets were reduced to 50 dimensions using PCA.

of optimizations to minimize data transfers and rendering costs to ensure interactivity.

3.2 Features and Clustering

To address challenge **C1**, VOCAL automatically extracts and learns features from the video segments to create the clusters used in the previous section. There are different ways to effectively pick these features for videos from an arbitrary domain. We consider three types of features that exhibit different tradeoffs between extraction speed and quality for clustering and modeling. The fastest set of features to extract are *basic features*. Basic features consist of color and motion statistics extracted from the raw pixel values of the videos. These features can be easily extracted with just a CPU. However, they are not specific to any one domain, and therefore their clustering and modeling performance may be limited. We next consider features extracted from *pretrained models*. While we do not expect pretrained models to exist for the specific domain being analyzed, it is possible that intermediate layers of neural networks trained on a different task and dataset may capture general information about the content of a video. These features are more expensive to extract than the basic features because they require a GPU to efficiently run inference on the pretrained model. The performance of these features will depend on both the pretraining dataset and the dataset being analyzed. For example, features extracted from a model trained on a self-driving car dataset would likely perform well on a different driving dataset, but features extracted from a model pretrained on stationary video data may perform poorly on egocentric video data.

Finally, we consider features extracted via *self-supervised learning*. Self-supervised learning learns embeddings for a dataset without labels. This is the most computationally expensive of the techniques we consider because it requires training a model before features can be extracted. Depending on the size of the model, this could require multiple GPUs to be done efficiently. However, self-supervised features are the most domain-specific because the embedding is learned from the unlabeled dataset itself. This could lead to superior performance on clustering and modeling tasks.

Given these different feature sets, the question then becomes when are the cheap basic features sufficient, and when should VOCAL invest the computational resources to learn more domain-specific features? We propose to always initialize exploration with the basic features because they are the cheapest to extract. This enables the user to begin interacting with their dataset without waiting for an expensive feature generation step. Then, VOCAL periodically extracts various feature representations from a catalog of pretrained models to determine whether any of them lead to clusters that better agree with the user-provided feedback than the basic features. During idle time when the user is not active, VOCAL learns self-supervised features. It picks a model size (e.g., ResNet18 vs. ResNet50 [27]) based on the compute resources available.

When clustering video segments using these features, in many cases the number of clusters is not known. This makes it challenging to use clustering algorithms that require the number of clusters to be specified, such as K-means. In general, it is better to specify more clusters than the expected number of labels because it is possible that a single label is best represented by multiple clusters. Alternatively, when the number of clusters is unknown it may be beneficial to use a clustering algorithm that can infer the number of clusters, such as a Variational Bayesian Gaussian mixture [13].

Preliminary results: To validate that it is possible to partition the video segments into clusters of similar activities, we use t-SNE [48] to visualize the distribution of a subset of video segments from a single deer containing different activities in Figure 6. We use the same basic color and motion features as in Section 2 (Figure 6a), features extracted from a pretrained Rubiksnet model [19] (Figure 6b), and features learned via self-supervised learning using BYOL [24] (Figure 6c). The figure shows that some activities are more easily partitioned (e.g., bedded and traveling are better separated than foraging and traveling), and the different feature sets perform similarly well. The results for other deer show similar trends.

To assess the clustering quality, we further perform clustering using a mixture model on video segments showing bedded, traveling, and foraging activities for ten deer, as shown in Figure 7. We use PCA to reduce all features down to 50 dimensions. We measure the homogeneity of the clusters, which is 1.0 when each cluster contains a single activity type. Higher homogeneity values indicate that video segments that represent the same activity belong to the same clusters. We measure homogeneity for exactly three clusters

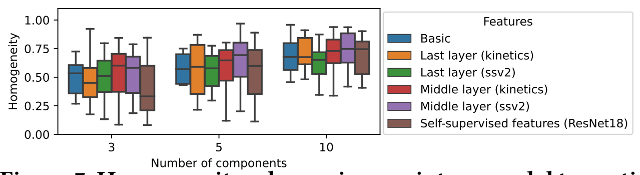


Figure 7: Homogeneity when using a mixture model to partition video segments into groups showing bedded, traveling, and foraging deer. The distribution is over video segments from ten deer.

(one per activity-type), as well as five and ten clusters to capture activities that naturally form multiple clusters. Interestingly, none of the feature sets leads to significantly better clustering. The self-supervised features were learned over frames sampled from the video segments. We expect that learning features over the video itself rather than individual frames would improve performance [21]. We observe that increasing the number of clusters is beneficial because it improves homogeneity.

We repeat the experiment for three deer that had an even larger number of data points, and we find the same trends with nearly identical homogeneity values. We also tried using simple K-means and achieved nearly identical results indicating that choosing a good number of clusters is more important than the cluster shape to achieve good homogeneity values. These results also show that the initial clustering quality leaves room for improvement, thus motivating the iterative user feedback approach from the previous section.

While we have focused on a single dataset and use case so far, we expect the initial techniques described in this section can be extended to achieve desired clusterings for any dataset with clearly identifiable activities or environmental conditions. We may need to expand our feature set or experiment with different segment durations in order to generalize.

4 COMPOSITIONAL QUERY PROCESSING

Once the interactive exploration and labeling process has trained domain-specific models that can detect simple activities and objects, users can start to execute compositional queries in VOCAL by utilizing these models and optionally other general-purpose models (e.g., YOLO [42]), as shown in Figure 2c. A compositional query consists of multiple spatially and temporally related objects (defined in more detail in Section 4.1). In this section, we change our running example from the deer dataset to a simpler scenario of traffic cameras capturing interactions between cars and pedestrians. While scientists are interested in compositional events over the deer dataset, we are still at the phase of data cleaning and initial exploration for that dataset.

Example: Compositional queries are helpful when monitoring and managing transportation in a city. As a concrete example, consider a scenario where a city planner seeks to determine if letting cars turn right at red lights creates hazards for pedestrians. In this application, the user may want to execute the following query: "Find all instances where a pedestrian is crossing an intersection while a car is turning at the same intersection." To find the target event, the user could: (i) apply some off-the-shelf model that can directly detect the whole event, (ii) detect object instances in each frame, and then explicitly define the event rule using objects and relationships between them, or (iii) manually examine randomly-sampled video segments until all instances are found. However, these approaches present the following challenges.

Challenges: The first challenge (**C1**) of compositional query processing is expressing the query. There are implicit constraints the user may fail to specify (e.g., the pedestrian should cross at the same location where the car is turning), and there are many satisfying spatial and temporal relationships (e.g., the pedestrian could be crossing at any corner of the intersection and the car could be turning before or after the pedestrian crosses). It can be difficult and tedious to specify all possible configurations for a target event.

The second challenge (**C2**) is the lack of pretrained models that can detect the complete compositional event. Unlike processing simple video queries where models trained on the target objects or actions may already exist (e.g., Faster R-CNN [43] for object detection, SlowFast network [20] for activity recognition) and can be applied directly to evaluate the query, such models are unlikely to exist for compositional queries that involve multiple objects and activities. The domain-specific models trained from the previous step are useful to identify entities or simple activities, but they are not sufficient for compositional queries. Additionally, it may not be possible to train a customized model to detect the target event because training such a model requires a large number of high-quality training examples, which may be absent from the videos, or even if they do exist, would require a tedious annotation process.

The final challenge (**C3**) we consider is the efficient execution of compositional queries. Because compositional queries concern objects and their relationships, a typical workflow for compositional query processing is to first run object detectors frame-by-frame and then match relationships [23, 37]. However, detecting every object in every frame is known to be prohibitively expensive [43].

Requirements: Given the challenges above, we seek for VOCAL to support users in finding user-defined compositional events without them writing explicit and accurate query specifications. VOCAL should be able to learn target query specifications through user feedback, while minimizing the user's labeling effort. VOCAL should also efficiently execute queries.

4.1 Query Definition

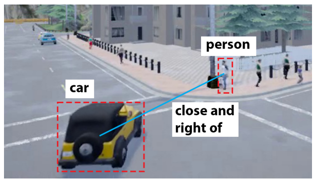
VOCAL uses the following data model for compositional queries.

Data Model: We define a video as a sequence of N consecutive frames $\{f_1, \dots, f_N\}$. The visual content of each frame f_i is fully described by a *scene graph* $g_i = (V_i, E_i)$ that consists of a set, V_i, of *all* objects in frame f_i, along with a set, E_i, of *all* relationships between those objects. Relationships can be semantic [36] (e.g., a person can *hold* an item) or spatial [30] (e.g., a car can be *near* a pedestrian). A *region graph*, g_{ij}, is a subgraph of a scene graph, i.e., $g_{ij} \subseteq g_i$.

Query Formulation: A compositional query q specifies an event of interest. The specification consists of the following elements:

(1) A temporally ordered sequence of region graphs $\{g_1, \dots, g_k\}$, which specifies that the event of interest consists of g_1, followed by g_2, followed by g_3, etc., where each g_i occurs one or multiple times.
(2) A set of predicates, which can be applied to region graphs, objects, or relationships.

(a) Negative frame



(b) Annotated positive frame (zoomed in for readability)

Figure 8: Sample frames from the Visual Road benchmark.

(3) A set of duration constraints, indicating that g_i should persist for at least (or at most, exactly, etc.) some number of consecutive frames before transitioning to the next region graph g_{i+1}.
(4) A window specification, expressed as a maximum number of frames that can separate g_1 and g_k.

Example: The example in Section 4 has two possible configurations: the pedestrian can be preceded or followed by the turning car. Using our semantics, one of the possible specifications is $q = \{g_1, g_2\}$, where g_1 contains one object, either a pedestrian or a car, and g_2 contains one object, either a pedestrian or a car, but a different object class than g_1. Further, there are predicates that both the pedestrian and the car must be in the intersection and the car must be turning.

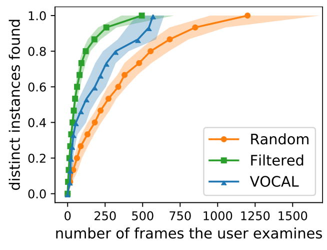
4.2 Iterative Query Refinement

To address challenges **C1** and **C2**, we advocate for an iterative query refinement approach. The user should not have to precisely express their query using the data model above. Instead, the user only needs to specify a simple initial query and then provide continual feedback to the system. VOCAL should learn the target event specification, while minimizing the user's video labeling effort.

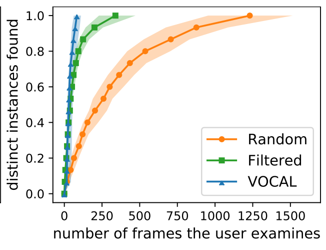
Approach: VOCAL first asks users for an initial, partial query for their target event. This query can be as simple as specifying only the target object classes (e.g., "car" and "pedestrian"), or it can include additional predicates (e.g., that the cars and pedestrians should be in the intersection). VOCAL then samples video frames that satisfy the partial query and asks the user to indicate if the frames are positive or negative examples of their target event.

There are various types of feedback that the user can provide beyond labeling frames as positive or negative examples. For instance, the user can specify important objects by drawing polygons around them. The user can also specify the relationships between objects that define the event. Further, the user can specify any objects or relationships that should not be present, such as "no bikes should be present at the intersection".

Although the user could express the target query using the data model introduced in Section 4.1, challenge **C1** describes how difficult it is. Instead, VOCAL hides the complexity and learns the query predicates by training a set of binary classifiers on the user feedback. VOCAL extracts proxy features that represent the event from different types of feedback, and incrementally trains classifiers to predict whether each frame is part of the target event and ranks the frames shown to the user by the classifiers' confidence. As the user labels more frames, the classifiers are retrained so that VOCAL gradually learns the target event specification.



(a) Without temporal selection



(b) With temporal selection

Figure 9: Fraction of distinct event instances found vs. number of examined frames. Solid curves show the median and shaded areas encompass the 25th to 75th percentiles. Each experiment is run 100 times.

One limitation of the approach described so far is that VOCAL risks showing the user multiple frames that belong to the same event. To address this challenge, we add a temporal selection heuristic that significantly lowers the probability of sampling a frame within a small time-window of a previously found positive example. Determining the right time-window and how to vary the sampling probability within it requires extra knowledge of the average duration and frequency of the target event, which can be either provided by the user or estimated by VOCAL over time.

Prototype Implementation Status: Currently, our prototype considers simple region graphs that consist of a set of objects, and we learn predicates over the spatial position of one of the objects, designated as the primary object. Our ongoing work is extending this early prototype.

While our design is broader, our prototype currently only utilizes features extracted from individual frames. An object detector cannot determine whether a car is *turning* based on a single frame. We observe, however, that spatial features are a reasonable proxy for many activities (e.g., turning cars appear more frequently at the corners of the intersection). Thus, for each labeled frame, VOCAL extracts five spatial features from the bounding box of one object that the user designates as primary (e.g., the turning car in our example): centroid coordinates ($c_x = (x_1 + x_2)/2, c_y = (y_1 + y_2)/2$), height ($h = y_2 - y_1$), width ($w = x_2 - x_1$) and aspect ratio ($r = w/h$), where x_1, y_1, x_2, y_2 are the upper-left and bottom-right coordinates of the bounding box. We do not yet leverage relative spatial positions of different objects.

As in the previous sections, for interactivity, VOCAL trains a random forest classifier rather than a DNN because it requires less labeled data and is faster to train. In our example, training a random forest classifier takes less than 0.03 seconds and requires only up to a few hundred labeled frames to find all instances of the event, as we show later in Figure 9, while training a specialized neural network is typically considerably more expensive. For example, the recently developed BlazeIt system requires 150,000 labeled frames and takes about 1 minute to train its specialized models [31].

Preliminary Results: We consider the example target query: "Find all instances where a car turns at an intersection *while* a pedestrian is crossing the same intersection." We say that VOCAL finds an instance of the target event if it finds at least one frame of that instance, since the user can later find all other adjacent frames of that instance by playing the video directly. In our experiment,

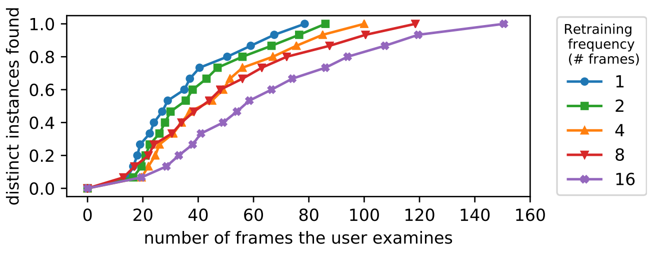


Figure 10: Retraining the classifier every 1, 2, 4, 8, and 16 annotations. Each experiment is run 100 times.

the user initializes the query by selecting the "car" and "person" classes from a list of object classes supported by the object detector and drawing a polygon to indicate the intersection region. For the user feedback, the user annotates positive frames to indicate which object instances are the important ones and whether their spatial configuration matters. We experiment with this query on a 15-minute VisualRoad [26] synthetic video captured at 25 fps and 960 × 540 resolution (Figure 8). To obtain the ground truth, we manually examine the video and label 951 frames as positive, which constitutes 15 distinct, non-consecutive instances of the target event. The average duration of the target event is 63 frames.

Figure 9 shows the cumulative fraction of distinct instances found as the user labels more frames. We compare our approach to two baselines: randomly sampling frames (`Random`) and randomly sampling frames that contain objects of interest within regions of interest (`Filtered`). In the figure, our approach is labeled as `VOCAL`, and we show the results with and without the temporal selection heuristic. In the experiment, we empirically set the window size of the heuristic to be four times the average duration of the target event, and we set the shape of the probability distribution to be a quadratic function. We also try other window sizes (e.g., 1, 2, 4, 6, 8 and 16 times the average duration) and shapes of the probability distribution (e.g. linear and quadratic), and we find the performance is quite robust. In both plots, `Random` performs poorly. Without the temporal selection heuristic (Figure 9a), `VOCAL` performs worse than `Filtered`. This is because `VOCAL` tends to choose frames with higher confidence scores and, as a result, shows many positive frames from already-found events to the user. With the heuristic (Figure 9b), `VOCAL` and `Filtered` find 60% of the positive instances similarly quickly. However, `VOCAL` outperforms `Filtered` by 3.6× for finding all positive instances.

Currently, VOCAL retrains the random forest classifier every time the user provides feedback because the model training time is negligible compared to user annotation time. However, this assumption would not hold if we switch to a more complex model or if the size of the training dataset grows larger. In such a case, we would retrain the model after some predefined amount of time, or use a heuristic to determine how many new annotations are needed to trigger a model retraining process. Moreover, model retraining could occur while the user is annotating or in the background, so that the user does not have to wait. Figure 10 shows how retraining frequency affects VOCAL's performance (with the temporal selection heuristic). The classifier is retrained after annotating every 1, 2, 4, 8, or 16 frames. The performance degrades slowly as the retraining frequency decreases, but is consistently better than the other two baselines (Figure 9).

4.3 Query Execution

A fundamental component of executing compositional queries is identifying the objects involved in the events. Executing DNN-based object detectors is known to be prohibitively expensive [43], and prior work includes methods that first filter frames using less expensive predicates [38] or build specialized models that can more efficiently answer a specific, simple query [31]. While compositional queries are more complicated than the queries considered by these prior systems because the former involves multiple objects with additional spatial constraints that may change over time, this additional complexity opens the door to several interesting optimizations, addressing challenge **C3**.

First, VOCAL can decompose a compositional event into multiple sub-events (e.g., a car turning event followed by a crowd of pedestrians walking event). VOCAL can try to estimate which sub-event is likely to be less frequent and find the rare event first, then process the remaining components of the compositional event on the filtered data only. To efficiently prune frames that do not contain sub-events, VOCAL can train specialized models for sub-events. For example, a specialized model could be trained to classify whether a frame contains a group of people in the intersection. Due to the complexity of the event, there are more ways it could be negative, and it is possible to build additional specialized models for these cases to filter out negative frames.

To efficiently prune out frames that do not contain sub-events, VOCAL can also try to determine which object in the sub-event, if any, can serve as a higher-selectivity predicate, and use existing techniques to find frames that contain that object. In our test video, 9.9%, 72.3%, and 99.7% of frames contain bikes, cars, and pedestrians, respectively. If the user issues a query involving both bicycles and cars, then VOCAL can apply a cheap probabilistic predicate on bikes during the first stage to filter out a large portion of frames that do not contain bikes, and then only invoke the object detector on remaining frames. This can even be done dynamically; VOCAL could start to run each object detector (with the less expensive predicates) in different parts of the video, measure the respective selectivity, and then switch to searching for the most selective object accordingly. While existing work orders filters for each object type to minimize the processing cost [51], we envision VOCAL going beyond this by incorporating user feedback to adjust the set of filters used to prune video frames.

Further, VOCAL can optimize the sampling rate. Prior work has optimized the sampling rate for object tracking queries [1], but with a different goal of reducing uncertainty rather than filtering out negative frames. As discussed in Section 4.2, VOCAL applies a temporal selection heuristic to sample fewer frames at locations where positive instances are already observed. We can do more, though, by incorporating knowledge of the duration of events. If the user specifies a lower bound on an event's duration, we can sample frames at this interval. If two sequential sampled frames do not contain the event, then all frames in between can be discarded. Finally, VOCAL can also reduce its sampling rate in parts of the video that do not encode much new information because these are less likely to contain new events. To do this, we can adapt research from the computer vision community that adaptively adjusts playback speed of a video based on its content [3], and ask queries over the speedup video.

Lastly, filtering in the compressed domain is also possible [50]. Even before decoding the video, the metadata (e.g., bitrate, frame size) or the compressed structure (e.g., quadtree, motion vectors) can be used for filtering. For instance, the presence of a group of objects within a specific region could result in a quadtree structure that uses more memory in this region to describe the details while the absence of objects would use less memory, so a filter based on this information can be devised.

We have shown that our initial prototype is effective for one dataset and one compositional query that involves objects and predicates over their spatial positions. We expect to extend our techniques to support a broader range of compositional queries on different datasets, which requires a more detailed definition of the data model and considering other types of user feedback. Further, we will look for other active learning techniques [17] to further reduce user effort, and other optimization techniques for more efficient query execution.

5 RELATED WORK

Prior work on recent VDBMSs focuses on accelerating specific types of queries, such as selection and aggregation queries [31], tracking queries [1], or on optimizing a single component of a VDBMS such as the storage layer [14, 25, 53]. However, these systems do not handle the full data processing pipeline. Panorama [56] enables users to specialize a vocabulary and make it more granular, but does not create new vocabularies for new domains (a goal of VOCAL).

Compositional queries: Caesar [37] and Rekall [23] support compositional queries but require all objects to be materialized beforehand, which is expensive. Further, Caesar assumes action detection models already exist, while Zeus [12] requires training new models and optimizes how actions are detected by using reinforcement learning techniques. Our query optimizations avoid expensive full materialization, do not require an action detection model to already exist or be trained, and automatically refine queries based on user feedback. TQVS [10] and Chen et al. [11] also detect and track all objects in every frame during the preprocessing stage and limit the types of compositional queries to temporal queries that count co-occurring objects. Vaas [2] enables users to execute a complex query by combining the output of simpler queries, but does not introduce any algebra to reason about spatiotemporal events, and SVQ++ [8] only supports spatial queries. However, we envision VOCAL supporting more general queries involving relationships between objects in space and time. Another line of work proposes various filters to facilitate compositional queries involving aggregation and spatial constraints [35], and interactions between objects [51], which could be applied to VOCAL to speed up query execution.

Semi-supervised clustering: Existing techniques generally either learn a distance metric based on constraints [52], modify the objective function with a penalty term for violated constraints, or employ a combination of these two strategies [4]. Contrary to our use-case where we begin with an unlabeled dataset, these existing techniques assume that all constraints are known ahead of time. Further, our goal is to update the clustering incrementally and rapidly so that the system is interactive, but these techniques require fully re-clustering the data when the constraints change. There exists an efficient greedy algorithm to incrementally incorporate new constraints [15]. However, it only adjusts the cluster assignments of data points that have been constrained. Adjusting the cluster assignments of only data points that have been constrained is not sufficient because the initial clustering of video segments is of unknown quality, so many segments may have to be reassigned which could lead to high manual effort and slow convergence.

Image clustering: QCluster [32] focuses on known-item search rather than fully partitioning the dataset. In contrast, our goal in the video exploration step is to use user feedback to learn domain-specific models rather than to locate video segments that match a particular query. Biswas and Jacobs [5] propose a technique that selects pairs of images for a user to compare to improve a clustering, which could potentially be incorporated into our active learning component. However, our optimization goal is different because rather than optimizing for the quality of the clusters themselves, VOCAL optimizes for the quality of the domain-specific models trained based on the clusters.

Video clustering: Cineast [44] and Vitrivr-Explore [29] are video browsing systems that are optimized for known-item search rather than general exploration and require the user to specify which features to use and how to weigh them. Near duplicate video retrieval techniques [34] assume the similarity between videos is known which is not true for a new dataset. A technique based on unsupervised clustering was developed for the specific application of detecting activities in egocentric sports videos [33], however it only uses motion features. This may not be sufficient for all types of events or environments, which is the reason VOCAL incorporates self-supervised features.

Self-supervised learning: Self-supervised learning can be used to learn embeddings from unlabeled data. SimCLR [9] demonstrated useful augmentation techniques to learn good embeddings on images, and BYOL [24] achieved similar performance without requiring negative pairs by learning to predict the representation of an augmented view of the input that is encoded with weights that are a moving average of the target model. This technique has been extended beyond images to learn features from videos [21]. The key challenge in incorporating these techniques into the data exploration process is deciding when it is most beneficial to invest the computational resources required to learn these representations. Users may prefer to use less computationally expensive features if they achieve similar performance.

Weak supervision: Rather than training domain-specific models using clusters refined by user-provided labels and constraints, weak supervision could instead be used to produce training data via data labeling functions as proposed in Snorkel [41]. Manually writing labeling functions over video data is challenging, so the user-labeled data could be passed to a system like Snuba [49] that automatically learns heuristics given a small amount of labeled data. However, it is still a challenge to decide which data the user should label. Labeling functions over video data can utilize rich temporal dependencies; the FlyingSquid framework [22] introduces techniques that enable weak supervision to efficiently operate over this type of label source.

6 CONCLUSION

In this paper, we presented a vision with preliminary experiments for VOCAL, a VDBMS that supports user-defined applications and compositional queries for any dataset. We proposed techniques to enable data cleaning and exploration over video data, even when no pretrained models exist to detect objects and events. We also introduced an iterative query refinement technique to work around the difficulty of expressing compositional queries, as well as ways to optimize query execution.

ACKNOWLEDGMENTS

This work was supported in part by NSF CCF-1703051, an award from the University of Washington Reality Lab, and a gift from the Intel Science and Technology Center for Big Data. Deer video data collection was carried out under National Science Foundation grants DEB1145902 to A.J. Wirsing and DEB1145522 to M.R. Heithaus and under UW IACUC Protocol #4226-01. Additional funding was provided by the Safari Club International Foundation, Conservation Northwest, the Washington Department of Fish and Wildlife (Aquatic Lands Enhancement Account (ALEA)), and the University of Washington Student Technology Fee (STF) program. We are grateful to the Colville Tribes Fish and Wildlife Department, and especially E. Krausz and R. Whitney, for permission to access their lands, guidance, and logistical support. We also thank the Okanogan–Wenatchee National Forest for logistical support.

REFERENCES

[1] Favyen Bastani, Songtao He, Arjun Balasingam, Karthik Gopalakrishnan, Mohammad Alizadeh, Hari Balakrishnan, Michael J. Cafarella, Tim Kraska, and Sam Madden. 2020. MIRIS: Fast Object Track Queries in Video. In *SIGMOD*. 1907–1921.
[2] Favyen Bastani, Oscar R. Moll, and Samuel Madden. 2020. Vaas: Video Analytics At Scale. *PVLDB* 13, 12 (2020), 2877–2880.
[3] Sagie Benaim, Ariel Ephrat, Oran Lang, Inbar Mosseri, William T. Freeman, Michael Rubinstein, Michal Irani, and Tali Dekel. 2020. SpeedNet: Learning the Speediness in Videos. In *CVPR*. 9919–9928.
[4] Mikhail Bilenko, Sugato Basu, and Raymond J. Mooney. 2004. Integrating constraints and metric learning in semi-supervised clustering. In *ICML*, Vol. 69.
[5] Arijit Biswas and David W. Jacobs. 2012. Active image clustering: Seeking constraints from humans to complement algorithms. In *CVPR*. 2152–2159.
[6] G. Bradski. 2000. The OpenCV Library. *Dr. Dobb's Journal of Software Tools* (2000).
[7] João Carreira and Andrew Zisserman. 2017. Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset. In *CVPR*. 4724–4733.
[8] Daren Chao, Nick Koudas, and Ioannis Xarchakos. 2020. SVQ++: Querying for Object Interactions in Video Streams. In *SIGMOD*. 2769–2772.
[9] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. 2020. A Simple Framework for Contrastive Learning of Visual Representations. In *ICML*, Vol. 119. 1597–1607.
[10] Yueting Chen, Xiaohui Yu, and Nick Koudas. 2020. TQVS: Temporal Queries over Video Streams in Action. In *SIGMOD*. 2737–2740.
[11] Yueting Chen, Xiaohui Yu, Nick Koudas, and Ziqiang Yu. 2021. Evaluating Temporal Queries Over Video Feeds. In *SIGMOD*. 287–299.
[12] Pramod Chunduri, Jaeho Bang, Yao Lu, and Joy Arulraj. 2021. Zeus: Efficiently Localizing Actions in Videos using Reinforcement Learning. *CoRR* abs/2104.06142 (2021).
[13] Adrian Corduneanu and Christopher M Bishop. 2001. Variational Bayesian model selection for mixture distributions. In *AIStat*, Vol. 2001. 27–34.
[14] Maureen Daum, Brandon Haynes, Dong He, Amrita Mazumdar, and Magdalena Balazinska. 2021. TASM: A Tile-Based Storage Manager for Video Analytics. In *ICDE*. 1775–1786.
[15] Ian Davidson, S. S. Ravi, and Martin Ester. 2007. Efficient incremental constrained clustering. In *SIGKDD*. 240–249.
[16] Justin Dellinger, Carolyn Shores, Apryle Craig, Shannon Kachel, Michael Heithaus, William Ripple, and Aaron Wirsing. 2021. Predators reduce niche overlap between sympatric prey. *Oikos* (12 2021). https://doi.org/10.1111/oik.08628
[17] Kyriaki Dimitriadou, Olga Papaemmanouil, and Yanlei Diao. 2016. AIDE: An Active Learning-Based Approach for Interactive Data Exploration. *TKDE* 28, 11 (2016), 2842–2856.
[18] Zeyad Ali Sami Emam, Hong-Min Chu, Ping-Yeh Chiang, Wojciech Czaja, Richard Leapman, Micah Goldblum, and Tom Goldstein. 2021. Active Learning at the ImageNet Scale. *CoRR* abs/2111.12880 (2021).
[19] Linxi Fan, Shyamal Buch, Guanzhi Wang, Ryan Cao, Yuke Zhu, Juan Carlos Niebles, and Li Fei-Fei. 2020. RubiksNet: Learnable 3D-Shift for Efficient Video Action Recognition. In *ECCV*, Vol. 12364. 505–521.
[20] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. 2019. SlowFast Networks for Video Recognition. In *ICCV*. 6201–6210.
[21] Christoph Feichtenhofer, Haoqi Fan, Bo Xiong, Ross B. Girshick, and Kaiming He. 2021. A Large-Scale Study on Unsupervised Spatiotemporal Representation Learning. In *CVPR*. 3299–3309.
[22] Daniel Y. Fu, Mayee F. Chen, Frederic Sala, Sarah M. Hooper, Kayvon Fatahalian, and Christopher Ré. 2020. Fast and Three-rious: Speeding Up Weak Supervision with Triplet Methods. In *ICML*, Vol. 119. 3280–3291.
[23] Daniel Y. Fu, Will Crichton, James Hong, Xinwei Yao, Haotian Zhang, Anh Truong, Avanika Narayan, Maneesh Agrawala, Christopher Ré, and Kayvon Fatahalian. 2019. Rekall: Specifying Video Events using Compositions of Spatiotemporal Labels. *CoRR* abs/1910.02993 (2019).
[24] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Ávila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. 2020. Bootstrap Your Own Latent - A New Approach to Self-Supervised Learning. In *NeurIPS*.
[25] Brandon Haynes, Maureen Daum, Dong He, Amrita Mazumdar, Magdalena Balazinska, Alvin Cheung, and Luis Ceze. 2021. VSS: A Storage System for Video Analytics. In *SIGMOD*. 685–696.
[26] Brandon Haynes, Amrita Mazumdar, Magdalena Balazinska, Luis Ceze, and Alvin Cheung. 2019. Visual Road: A Video Data Management Benchmark. In *SIGMOD*. 972–987.
[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *CVPR*. 770–778.
[28] M Heithaus, L Dill, G Marshall, and B Buhleier. 2002. Habitat use and foraging behavior of tiger sharks (Galeocerdo cuvier) in a seagrass ecosystem. *Marine Biology* 140, 2 (2002), 237–248.
[29] Silvan Heller, Mahnaz Parian, Maurizio Pasquinelli, and Heiko Schuldt. 2020. Vitrivr-Explore: Guided Multimedia Collection Exploration for Ad-hoc Video Search. In *SISAP (Lecture Notes in Computer Science, Vol. 12440)*. 379–386.
[30] Jingwei Ji, Ranjay Krishna, Fei-Fei Li, and Juan Carlos Niebles. 2020. Action Genome: Actions As Compositions of Spatio-Temporal Scene Graphs. In *CVPR*. 10233–10244.
[31] Daniel Kang, Peter Bailis, and Matei Zaharia. 2019. BlazeIt: Optimizing Declarative Aggregation and Limit Queries for Neural Network-Based Video Analytics. *PVLDB* 13, 4 (2019), 533–546.
[32] Deok-Hwan Kim and Chin-Wan Chung. 2003. Qcluster: Relevance Feedback Using Adaptive Clustering for Content-Based Image Retrieval. In *SIGMOD*. 599–610.
[33] Kris Makoto Kitani, Takahiro Okabe, Yoichi Sato, and Akihiro Sugimoto. 2011. Fast unsupervised ego-action learning for first-person sports videos. In *CVPR*. 3241–3248.
[34] Giorgos Kordopatis-Zilos, Symeon Papadopoulos, Ioannis Patras, and Yiannis Kompatsiaris. 2017. Near-Duplicate Video Retrieval with Deep Metric Learning. In *ICCV*. 347–356.
[35] Nick Koudas, Raymond Li, and Ioannis Xarchakos. 2020. Video Monitoring Queries. In *ICDE*. 1285–1296.
[36] Ranjay Krishna, Vincent S. Chen, Paroma Varma, Michael Bernstein, Christopher Ré, and Fei-Fei Li. 2019. Scene Graph Prediction With Limited Labels. In *ICCV*. 2580–2590.
[37] Xiaochen Liu, Pradipta Ghosh, Oytun Ulutan, B. S. Manjunath, Kevin S. Chan, and Ramesh Govindan. 2019. Caesar: cross-camera complex activity recognition. In *SenSys*. 232–244.
[38] Yao Lu, Aakanksha Chowdhery, Srikanth Kandula, and Surajit Chaudhuri. 2018. Accelerating Machine Learning Inference with Probabilistic Predicates. In *SIGMOD*. 1493–1508.
[39] Sarah McQuate. 2020. UW researchers driving around Seattle to track COVID-19 response over time. https://www.washington.edu/news/2020/09/30/uw-researchers-drive-around-seattle-track-covid-19-response-over-time/.
[40] Parmita Mehta, Stephen Portillo, Magdalena Balazinska, and Andrew J. Connolly. 2020. Toward Sampling for Deep Learning Model Diagnosis. In *ICDE*. 1910–1913.
[41] Alexander Ratner, Stephen H. Bach, Henry R. Ehrenberg, Jason Alan Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid Training Data Creation with Weak Supervision. *PVLDB* 11, 3 (2017), 269–282.

[42] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. 2016. You Only Look Once: Unified, Real-Time Object Detection. In *CVPR*. 779–788.

[43] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. 2015. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *NeurIPS*. 91–99.

[44] Luca Rossetto, Ivan Giangreco, and Heiko Schuldt. 2014. Cineast: A Multi-feature Sketch-Based Video Retrieval Engine. In *ISM*. 18–23.

[45] Burr Settles. 2009. Active learning literature survey. (2009).

[46] Lifeng Shang, Linjun Yang, Fei Wang, Kwok-Ping Chan, and Xian-Sheng Hua. 2010. Real-time large scale near-duplicate web video retrieval. In *MM*. 531–540.

[47] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. 2018. A Closer Look at Spatiotemporal Convolutions for Action Recognition. In *CVPR*. 6450–6459.

[48] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *JMLR* 9, 11 (2008).

[49] Paroma Varma and Christopher Ré. 2018. Snuba: Automating Weak Supervision to Label Training Data. *PVLDB* 12, 3 (2018), 223–236.

[50] Chao-Yuan Wu, Manzil Zaheer, Hexiang Hu, R. Manmatha, Alexander J. Smola, and Philipp Krähenbühl. 2018. Compressed Video Action Recognition. In *CVPR*. 6026–6035.

[51] Ioannis Xarchakos and Nick Koudas. 2021. Querying for Interactions. *TKDE* (2021), 1–1.

[52] Eric P. Xing, Andrew Y. Ng, Michael I. Jordan, and Stuart J. Russell. 2002. Distance Metric Learning with Application to Clustering with Side-Information. In *NeurIPS*. 505–512.

[53] Tiantu Xu, Luis Materon Botelho, and Felix Xiaozhu Lin. 2019. VStore: A Data Store for Analytics on Large Videos. In *EuroSys*. 16:1–16:17.

[54] Piyush Yadav and Edward Curry. 2019. VidCEP: Complex Event Processing Framework to Detect Spatiotemporal Patterns in Video Streams. In *BigData*. 2513–2522.

[55] Shiyun Yang, Emily Bailey, Zhengye Yang, Jonatan Ostrometzky, Gil Zussman, Ivan Seskar, and Zoran Kostic. 2020. COSMOS Smart Intersection: Edge Compute and Communications for Bird's Eye Object Tracking. In *PerCom*. 1–7.

[56] Yuhao Zhang and Arun Kumar. 2019. Panorama: A Data System for Unbounded Vocabulary Querying over Video. *PVLDB* 13, 4 (2019), 477–491.