Vaas: video analytics at scale

Vaas: Video Analytics At Scale

Favyen Bastani
MIT CSAIL
favyen@csail.mit.edu

Oscar Moll
MIT CSAIL
orm@csail.mit.edu

Sam Madden
MIT CSAIL
madden@csail.mit.edu

ABSTRACT

We demonstrate Vaas, a video analytics system for large-scale datasets. Vaas provides an interactive interface to rapidly develop and experiment with different workflows for solving a video analytics task. Users express these workflows as Vaas queries, which specify data flow graphs where nodes may be implemented by machine learning models, custom code, or basic built-in operations (e.g., cropping, selecting detections of by class, filtering tracks by bounding boxes). For example, the problem of detecting lane change events in dashboard camera video could be solved directly as an activity recognition task, by training a model to classify whether a segment of video contains a lane change, or decomposed into a set of simpler tasks, such as detecting lane markers and then identifying shifts in the detected lanes. Our system interface incorporates a query composition tool, where users can rapidly compose operations to implement a workflow, and an exploration tool, where users can experiment with a query by applying it over samples from the dataset to fix bugs and tune parameters. Vaas incorporates recent work in approximate video query processing to support the fast, interactive execution of queries, and accelerates the annotation process of hand-labeling examples to train models by allowing users to annotate over the outputs of previously expressed queries rather than the entire video dataset.

PVLDB Reference Format:
Favyen Bastani, Oscar Moll, Sam Madden. Vaas: Video Analytics At Scale. *PVLDB*, 13(12): 2877-2880, 2020.
DOI: https://doi.org/10.14778/3415478.3415498

1. INTRODUCTION

Video data today is collected in vast quantities by cameras in a diverse range of settings, including traffic cameras, aerial drones, mobile phones, and dashboard cameras and semi-autonomous vehicles. Video analytics can provide enormous benefits to many applications: video of roads and junctions can inform traffic planning decisions [1], video captured from motor vehicles can provide insights into driving behavior [2], and videos shared publicly can indicate trends in online communities [3].

However, the unstructured nature of video results in two substantial costs in video analytics: the development cost of implementing analytics workflows (including both hand-labeling data for training machine learning models, and developing the data processing pipeline that incorporates those models), and the query execution cost to apply those models over large video datasets. Oftentimes, the enormity of these costs makes it unclear how to even begin performing an analytics task. For example, suppose a traffic planning analyst wishes to find segments of traffic camera video where a car runs a red light. Directly annotating these instances as an activity recognition problem, where a classifier is trained over segments of video, likely makes the task intractable, as it may involve manually watching hours of video to label just one run-red-light instance. Similarly, achieving high recognition accuracy likely involves training a deep neural network, and applying such a model on thousands of hours of video is expensive.

To address these challenges, we demonstrate Vaas, a system for video analytics at scale. Vaas provides an interactive interface that enables users to efficiently explore different approaches to solve an analytics task. Additionally, Vaas incorporates recent work in approximate video query processing techniques, including probabilistic predicates [4, 5, 6] and variable framerate tracking [7], as well as an incremental query execution architecture, to ensure that interface interactions are fast and to rapidly execute queries over entire video datasets once users finalize their queries.

Vaas queries are data flow graphs that define multi-stage execution pipelines. Nodes in a data flow graph may be models, which learn parameters from hand-labeled examples, or Python functions, which can inspect video directly or operate over the outputs of models or other functions. This architecture allows the flexible expression of hybrid queries that combine models with code to solve an analytics task with minimal effort. In the example run-red-light task, an analyst may decide to split up the task into two components: find cars that pass straight through a junction in a certain direction, and identify times when a light that controls traffic in that direction is red. For the first component, the analyst may leverage a pre-trained object detection model to detect cars, and a heuristic multi-object tracker to compute the trajectories of cars through video. Then, the analyst can program a function that inputs the car tracks, and selects only the tracks that correspond to the direction of interest. For the second component, the analyst

Proceedings of the VLDB Endowment, Vol. 13, No. 12
ISSN 2150-8097.
DOI: https://doi.org/10.14778/3415478.3415498

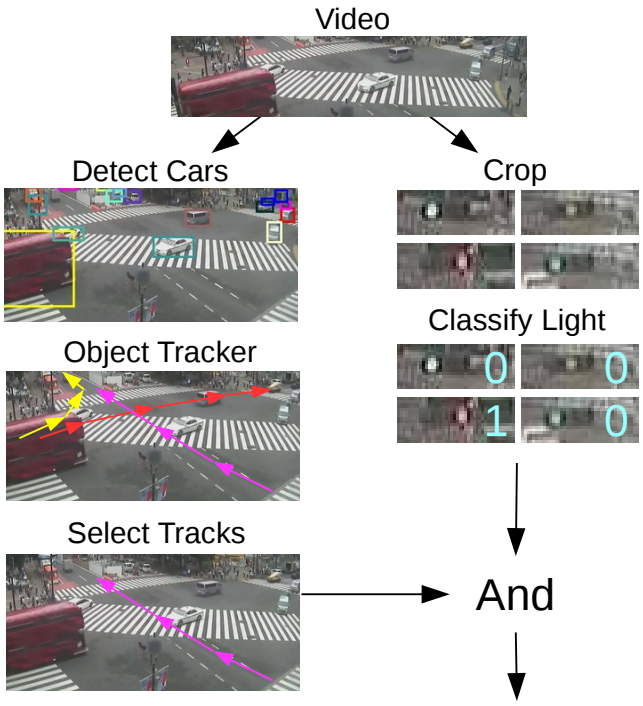

Figure 1: An example data flow graph for selecting cars that run a red light.

may begin by cropping the video to extract a small window around a traffic light. The analyst may then attempt to program a function that directly inputs the cropped video and processes the per-pixel hues to estimate the traffic light state. However, doing so may require substantial effort to ensure the processing is accurate at different times of day across the traffic camera dataset, and so the analyst may find it faster to simply annotate the light state of several randomly sampled cropped video frames and train a classification model on the annotations. Finally, the analyst can program a function to input the selected car tracks and inferred light states, and output cars that run red lights by only selecting tracks where the light state is red for the duration of the track. We show the data flow graph for this example in Figure 1.

We demonstrate Vaas by applying it on various analytics tasks, including tasks over dashcam video in the Berkeley DeepDrive dataset [8] and over traffic camera footage in the YTStream dataset [7].

2. SYSTEM INTERFACE

We show the Vaas system interface in Figure 2. The interface consists of three tools: the annotation tool (Section 2.2) focuses on supporting the hand-labeling of video data for training machine learning models; the query composition tool (Section 2.3) enables users to express a particular approach for solving an analytics task (i.e., a Vaas query); and the exploration tool (Section 2.4) enables users to experiment with and improve their analytics workflow implementations.

2.1 Data Types

In general, video analytics may involve several data types. The run-red-light example involved cropping video, processing car tracks, and classifying traffic light states. Vaas supports four data types:

- Video: represents not only video from the underlying dataset being analyzed, but also intermediate segmentation and other image-based model outputs.
- Detections: represents object detections, e.g., bounding boxes of cars detected in video. Detections are extracted in individual video frames, and may be bounding boxes, points, lines, or polygons.
- Tracks: represents sequences of detections corresponding to the same object instance, e.g., the trajectory of a car through the camera frame over the segment of video in which it is visible.
- Classes: represents classifier outputs, for both image classification tasks (select frames that contain ambulances) and activity recognition tasks (select video segments where cars run red lights).

Vaas represents data as time series, where each timestep is a video frame. For example, detection and track data associate a list of detections with each frame.

2.2 Annotation

The annotation tool enables efficient hand-labeling of detections, tracks, and classes in video. Users can export their annotations to standard formats for training object detectors, object classifiers, image classifiers, and activity recognition models. The trained models can then be linked back into Vaas so that they can be used as a node in a data flow graph. Detections and tracks are labeled through a click-based interface, where users draw shapes overlayed on video frames. Classes are labeled through a button-based interface, and class annotations may be labeled over individual frames (image classification) or over video segments (activity recognition).

Oftentimes, though, users are interested in rare events that are tedious to annotate. For example, a driving behavior researcher may be interested in finding segments of dashcam video where another car overtakes the recording car and suddenly brakes. These overtake-and-brake events likely happen infrequently; thus, if the annotation tool provides examples for labeling by uniformly sampling video segments from the entire video dataset, it may take a long time to label even one positive example.

Instead, Vaas supports performing annotation in the context of a query: rather than label all video, if the user can express a low-precision, high-recall query that approximates their analytics task, then the user can opt to focus annotation on segments of video where this query produces positive outputs. For overtake-and-brake events, the user may leverage a car detector and tracker to express a query that selects car tracks that begin on the leftmost portion of the camera frame, and end in the middle. This corresponds to a car overtaking on the left, and most overtake-and-brake events likely satisfy the query (high-recall). On the other hand, there may be other situations, such as cars passing without changing into the recording car's lane, that also

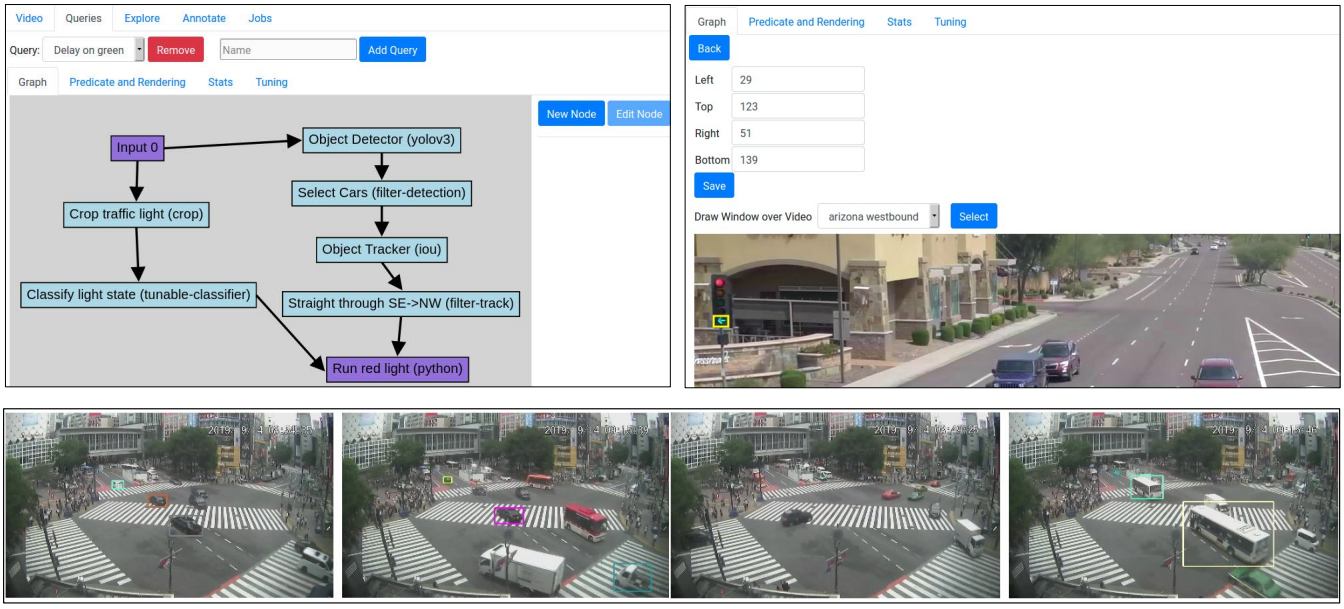

Figure 2: The Vaas system interface. At top-left, the query composition tool enables users to compose machine learning models, custom code, and built-in operations to implement analytics workflows. At top-right, we show the editor for a node using the built-in crop operation. At bottom, we show the exploration tool, where users can interactively experiment with their queries and visualize the query outputs; here, Vaas renders four videos for the executed query.

satisfy the query (low-precision), so the query does not fully solve the analytics task. Nevertheless, the user can initialize the Vaas annotation tool under this query to accelerate the hand-labeling process: the frequency of overtake-and-brake events is far higher over the query outputs than when sampling uniformly across the dashcam dataset. After training an activity recognition model on the annotations, the user can revise their query to apply a logical conjunction (AND) of their original query and the model.

2.3 Composition

The query composition tool enables users to rapidly implement various analytics workflows. A query specifies a directed acyclic graph, where source nodes correspond to the underlying video dataset, and information trickles down to one or more sink nodes (Figure 1) that Vaas outputs to disk. Each non-source node may be implemented by a machine learning model, custom Python code, or a built-in operation.

To make it easier for users to incorporate models into a workflow, Vaas provides a library of built-in models, but users may link their own models if desired. When using built-in models, which include YOLOv3 and a shallow classification CNN, users can either directly apply a model pre-trained on COCO or ImageNet, or fine-tune the model on their own set of annotations. Models typically input raw video, but may also input data computed through other nodes, such as object tracks. Similarly, models are often deep neural networks, but (especially on non-video inputs) can use other techniques such as support vector machine, nearest neighbor classification, and logistic regression.

Custom Python code consists of Python functions that can take inputs from one or more other nodes, and can be implemented directly in the composition interface. Vaas provides a Python library to make it easy for users to express a diverse range of queries. For example, the library provides functions to reason about spatial and temporal constraints relevant to detections, tracks, and classification outputs.

2.4 Exploration

Users can experiment with queries by interactively evaluating their outputs through the exploration tool: when a user wishes to test a query, Vaas populates a scrollable interface with outputs of the query by executing it incrementally over the video dataset. In the interface, users can visualize different data types; for example, for tracks, Vaas by default displays bounding boxes in output tracks overlayed on the underlying video data, with distinct instances shown in different colors. Thus, if a query is not working as expected, the user can debug the issue using the video outputs rendered by the exploration tool, and revise the query in the composition tool. Once a user is satisfied with the query's performance, the query can be executed over the entire dataset.

3. QUERY EXECUTION ENGINE

We show the architecture of the Vaas query execution engine in Figure 3. Approximate query processing (AQP) optimizations are implemented as plugins in the execution engine, and Vaas automatically selects a subset of optimizations to apply for a given query. Additionally, when Vaas ingests a video, it lazily produces copies of the video at reduced resolutions and framerates; Vaas may apply models required as inputs to the query, or auxiliary models trained by AQP optimizations, over these copies to avoid expensive repeated decoding of the original full-resolution video.

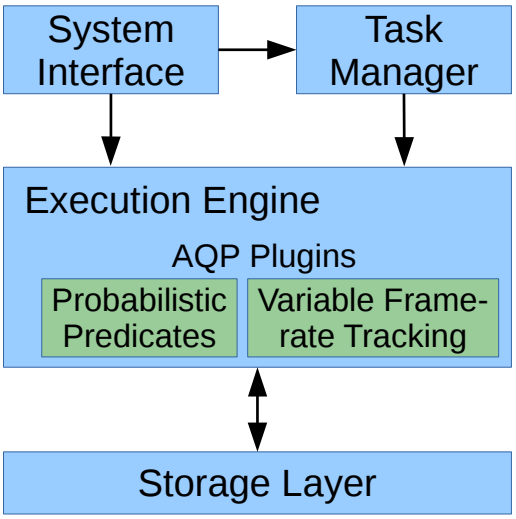

Figure 3: Vaas system architecture. The system interface submits interactive queries directly to the execution engine, while queries over full datasets are scheduled by the task manager.

3.1 Approximate Query Processing

Vaas incorporates two recent approximate query processing techniques to accelerate query execution.

Probabilistic predicates [4, 5, 6]. This optimization trains a fast, weak classifier to approximate outputs on image classification and activity recognition tasks, and applies the classifier as a low-precision, high-recall filter. Initially, rather than apply the full execution pipeline specified by the query, we apply the weak classifier and collect its confidence scores across segments of the video dataset. If the confidence score is sufficiently small in a segment, we prune the segment without further processing. We apply the full pipeline in the remaining segments to ensure high precision in the query outputs.

Vaas performs validation in randomly sampled video segments to automatically determine the type of classifier to train (e.g., support vector machine, or shallow neural network), the confidence threshold for pruning, and the input resolution.

Variable framerate tracking [7]. This optimization addresses queries involving object tracks. Instead of performing tracking at the full video framerate, we track objects at substantially reduced framerates when doing so only negligibly impairs accuracy. The execution engine automatically increases the framerate at which it samples video when needed to maintain high-accuracy, such as in busier segments of video where there is a higher density of object instances.

3.2 Storage

The storage layer maintains data produced during query execution that may be re-used in other analytics tasks. This includes copies of video at reduced resolutions and framerates, and also the outputs of intermediate models used in a query. Vaas accounts for data already populated in the storage system when developing an execution plan for a new query over an existing dataset.

4. DEMONSTRATION SCENARIOS

We demonstrate Vaas on several analytics tasks over two video sources. First, over dashcam video in the Berkeley DeepDrive dataset [8], tasks include detecting overtake-and-brake events, detecting instances when the recording car changes lanes, and detecting situations not labeled in typical object detection datasets such as camera glare and construction zones. Second, over traffic camera video in the YTStream dataset [7], tasks include counting left-turning cars, finding cars that run red lights, and finding cars that stop in the crosswalk.

5. REFERENCES

[1] Zheng Tang, Milind Naphade, Ming-Yu Liu, Xiaodong Yang, Stan Birchfield, Shuo Wang, Ratnesh Kumar, David Anastasiu, and Jenq-Neng Hwang. CityFlow: A City-Scale Benchmark for Multi-Target Multi-Camera Vehicle Tracking and Re-Identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8797–8806, 2019.

[2] Lex Fridman, Daniel E Brown, Michael Glazer, William Angell, Spencer Dodd, Benedikt Jenik, Jack Terwilliger, Aleksandr Patsekin, Julia Kindelsberger, Li Ding, et al. MIT Advanced Vehicle Technology Study: Large-Scale Naturalistic Driving Study of Driver Behavior and Interaction with Automation. *IEEE Access*, 7:102021–102038, 2019.

[3] Yu-Han Tiffany Chen. *Interactive Object Recognition and Search over Mobile Video*. PhD thesis, Massachusetts Institute of Technology, 2017.

[4] Daniel Kang, John Emmons, Firas Abuzaid, Peter Bailis, and Matei Zaharia. NoScope: Optimizing Neural Network Queries over Video at Scale. In *PVLDB*, 10(11):1586–1597, 2017.

[5] Yao Lu, Aakanksha Chowdhery, Srikanth Kandula, and Surajit Chaudhuri. Accelerating Machine Learning Inference with Probabilistic Predicates. In *International Conference on Management of Data (SIGMOD)*, pages 1493–1508. ACM, 2018.

[6] Daniel Kang, Peter Bailis, and Matei Zaharia. Challenges and Opportunities in DNN-Based Video Analytics: A Demonstration of the BlazeIt Video Query Engine. In *Conference on Innovative Data Systems Research (CIDR)*, 2019.

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

[8] Huazhe Xu, Yang Gao, Fisher Yu, and Trevor Darrell. End-to-end Learning of Driving Models from Large-scale Video Datasets. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2174–2182, 2017.