DATA ANALYTICS USING DEEP LEARNING

GT 8803 // SIDDHARTH BISWAL

LECTURE #03:BLAZEIT: FAST EXPLORATORY VIDEO QUERIES USING NEURAL NETWORKS

CREATING THE NEXT®

TODAY'S PAPER

- BlazeIt: Fast Exploratory Video Queries using Neural Networks
  - Daniel Kang, Peter Bailis, Matei Zaharia
- Slides inspired from on a presentation by Daniel Kang for NoScope Paper

TODAY'S AGENDA

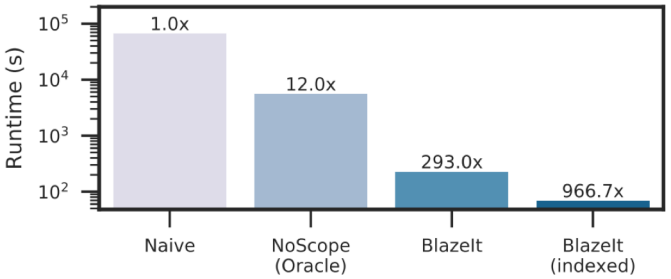
- Problem Overview
- Key Idea
- Technical Details
- Experiments
- Discussion

INTRODUCTION

- With video volume growth, deep learning has become solution of choice for analytics
- But deep learning methods are 10× slower than real time (3 fps) on a $8,000 GPU: Not scalable
- BLAZEIT: a system that optimizes queries over video for **spatiotemporal information of objects**.

INTRODUCTION

- Queries FRAMEQL, a declarative language for exploratory video analytics, that enables video-specific query optimization
- Authors use control variates to video analytics and provide advances in specialization for aggregation queries.
- Importance-sampling using specialized NNs for cardinality-limited video search (i.e. scrubbing queries).
- Third, we show how to infer new classes of filters for content-based selection.



USE CASES

BLAZEIT focuses on **exploratory queries**:

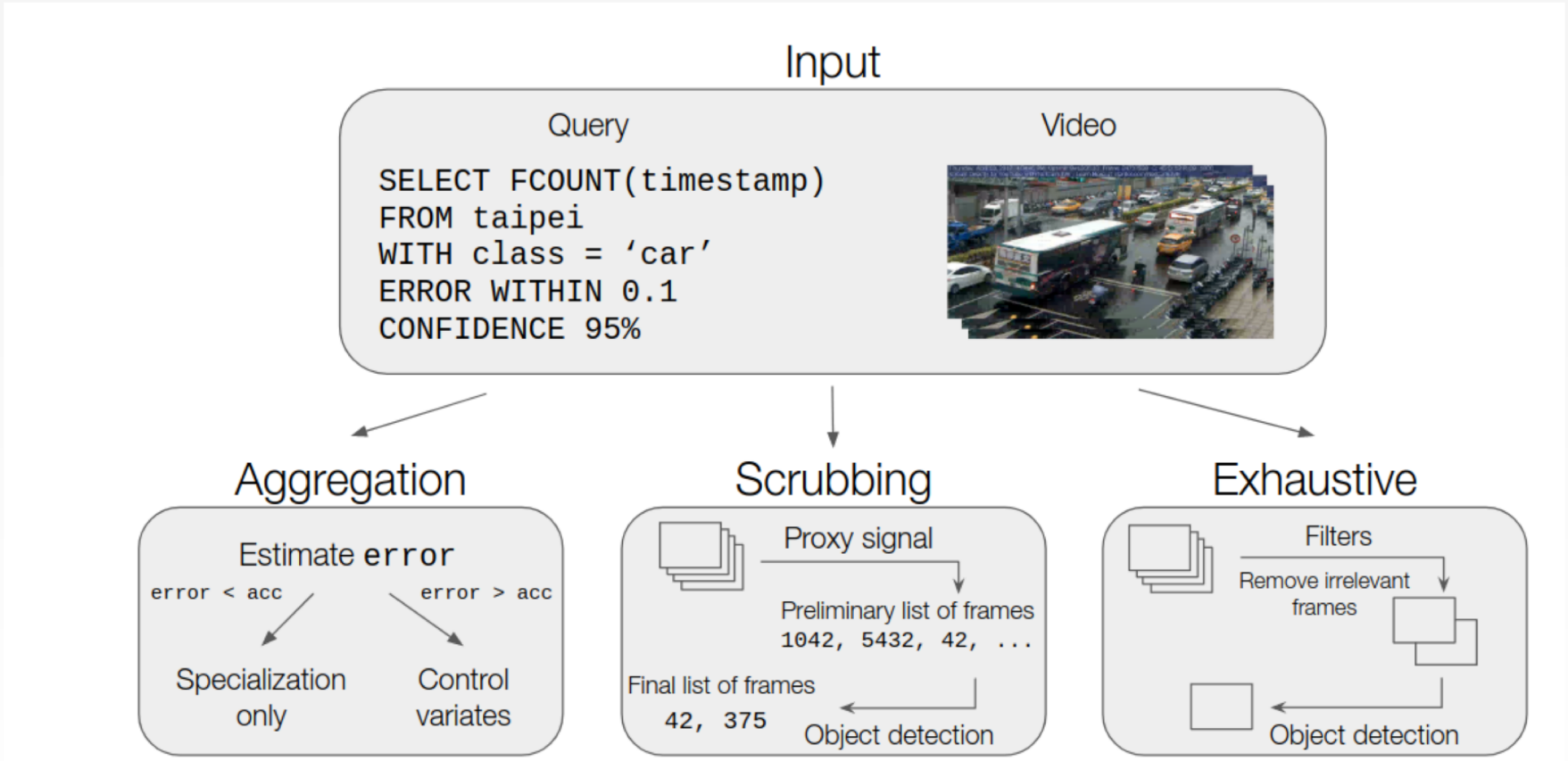
Queries that can help a user understand a video quickly, e.g., queries for aggregate statistics (e.g., number of cars) or relatively rare events (e.g., events of many birds at a feeder) in videos

1. Urban planning: Using traffic cameras perform traffic metering and determine which days and times are the busiest.

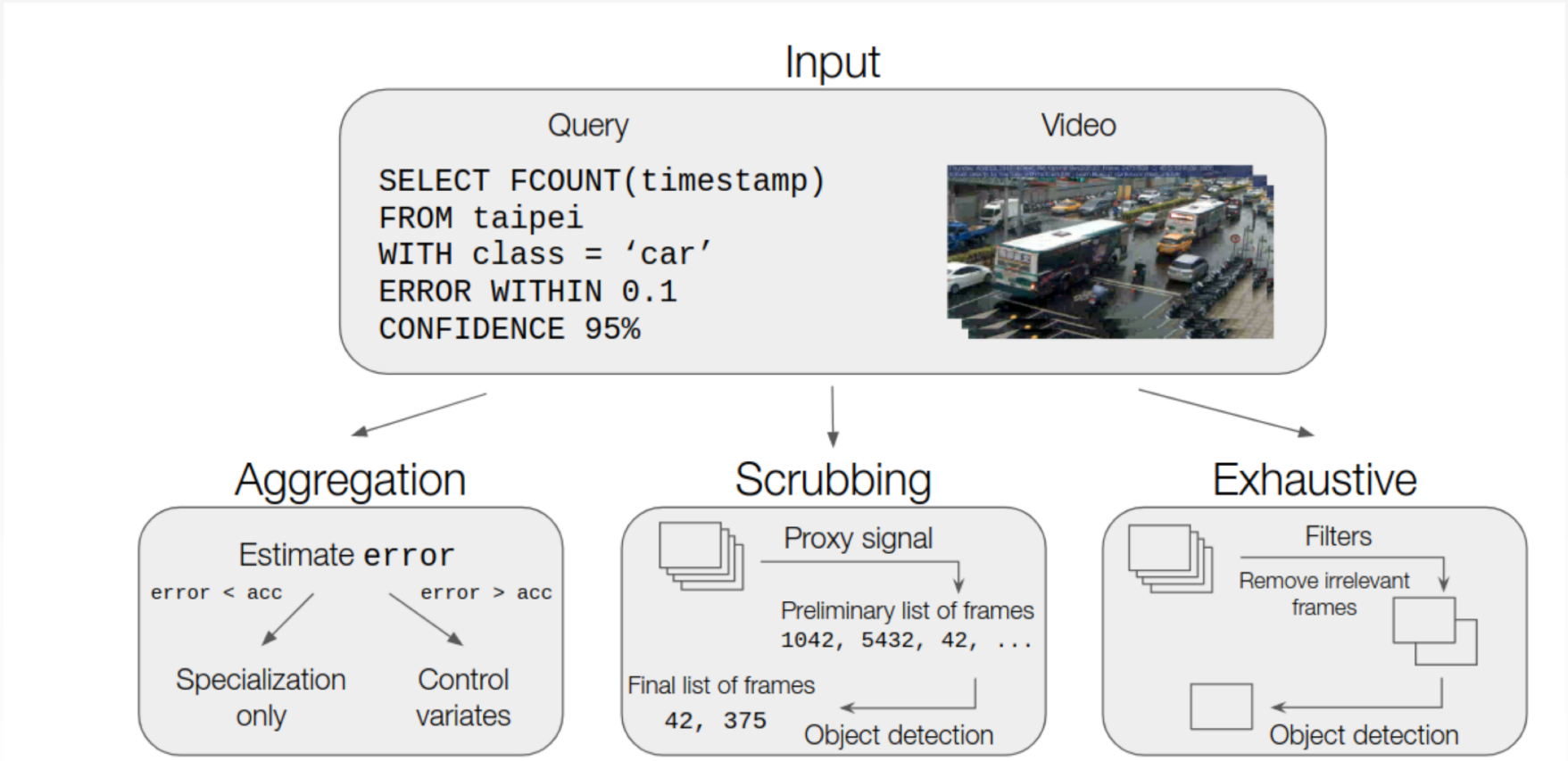
2. Autonomous vehicle analysis: anomalous behavior of the driving software given specific circumstances

3. Store planning: retail store owner places a CCTV in the store. Analytics can be use to segment the video into aisles and counts the number of people that walk through each aisle to understand which products are popular and which ones are not. Hence this information can be used for planning store layout, aisle layout, and product placement.

SYSTEM OVERVIEW



SYSTEM OVERVIEW



FRAMEQL

- a SQL-like language for querying spatiotemporal information of objects in video
- 1. Encoding queries via a declarative language interface separates the specification and implementation of the system, which enables query optimization (discussed later)
- 2. As SQL is the lingua franca of data analytics, FRAMEQL can be easily learned by users familiar with SQL and enables interoperability with relational algebra
- Input: video feed, Query: the frame-level content
  - specifically the objects appearing in the video over space and time by content and location
- FrameQL allows selection, projection, and aggregation of objects, and, by returning relations, can be composed with standard relational operators

DATA SCHEMA

- Data Schema for FrameQL

Field	Type	Description
`timestamp`	float	Time stamp
`class`	string	Object class (e.g., bus, car, person)
`mask`	(float, float)*	Polygon containing the object of interest, typically a rectangle
`trackid`	int	Unique identifier for a continuous time segment when the object is visible
`features`	float*	The feature vector output by the object detection method.

Table 1: FRAMEQL's data schema contains spatiotemporal and content information related to objects of interest, as well as metadata (class, identifiers). Each record represents an object appearing in one frame; thus a frame may have many or no records. The features can be used for downstream tasks.

FRAMEQL

- Additional syntactic elements in FRAMEQL

Syntactic element	Description
`FCOUNT`	Frame-averaged count. Equivalent to `COUNT(*) / MAX(timestamp)`. Also equivalent to a time-averaged count.
`ERROR WITHIN`	Absolute error tolerance
`FPR WITHIN`	Allowed false positive rate
`FNR WITHIN`	Allowed false negative rate
`CONFIDENCE`	Confidence interval
`GAP`	Minimum distance between returned frames

Table 2: Additional syntactic elements in FRAMEQL. Some of these were taken from BlinkDB.

FRAMEQL

```
SELECT FCOUNT(*)
FROM taipei
WHERE class = 'car'
ERROR WITHIN 0.1
AT CONFIDENCE 95%
```

(a) The FRAMEQL query for counting the frame-averaged number of cars within a specified error and confidence.

```
SELECT timestamp
FROM taipei
GROUP BY timestamp
HAVING SUM(class='bus')>=1
   AND SUM(class='car')>=5
LIMIT 10 GAP 300
```

(b) The FRAMEQL query for selecting 10 frames of at least one bus and five cars, with each frame at least 10 seconds apart (at 30 fps, 300 frames corresponds to 10s).

FRAMEQL

```
SELECT *
FROM taipei
WHERE class = 'bus'
  AND redness(content) >= 17.5
  AND area(mask) > 100000
GROUP BY trackid
HAVING COUNT(*) > 15
```

(c) The FRAMEQL query for selecting all the information of red buses at least 100,000 pixels large, in the scene for at least 0.5s (at 30 fps, 0.5s is 15 frames). The last constraint is for noise reduction.

```
SELECT COUNT (DISTINCT trackid)
FROM taipei
WHERE class = 'car'
```

which is not the same as counting the average number of cars in a frame, as this query looks for distinct instances of cars using `trackid` (cf. Figure 3a).

Second, error rates can be set using syntax similar to BlinkDB [5]:

```
SELECT COUNT(*)
FROM taipei
WHERE class = 'car'
ERROR WITHIN 0.1 CONFIDENCE 95%
```

Third, NOSCOPE can be replicated as FRAMEQL queries of the form:

```
SELECT timestamp
FROM taipei
WHERE class = 'car'
FNR WITHIN 0.01
FPR WITHIN 0.01
```

Finally, a UDF could be used to classify cars:

```
SELECT *
FROM taipei
WHERE class = 'car'
  AND classify(content) = 'sedan'
```

FRAMEQL

FrameQL: A Query Language for Complex Visual Queries over Video

Data: Training data, held-out data, unseen video, $uerr \leftarrow$ user's requested error rate, $conf \leftarrow$ user's confidence level
Result: Estimate of requested quantity
train specialized NN on training data;
$err \leftarrow$ specialized NN error rate on held-out data;
$\tau \leftarrow$ average of specialized NN over unseen video;
if $P(err < uerr) < conf$ **then**
  return τ;
else
  $\hat{m} \leftarrow$ result of control variates;
  return $\hat{m}$;
end

Algorithm 1: BLAZEIT's procedure to return the results of an aggregate query. This is performed when there are enough examples to train a specialized NN.

IMPLEMENTATION DETAILS

Video ingestion:
1. Loads the video using OpenCV,
2. Resizes the frames to the appropriate size for each model
3. Normalizes the pixel values appropriately

Specialized NN training:
We train the specialized NNs using PyTorch v0.4.
1..Video are ingested and resized to 65×65 pixels and normalized using standard ImageNet normalization .
2.Cross Entropy with batch size of 16.
3. SGD with a momentum of 0.9. Our specialized NNs use a "tiny ResNet" architecture, a modified version of the standard ResNet architecture [32], which has 10 layers and a starting filter size of 16.

Identifying objects across frames
1. Our default implementation for computing trackid use motion IOU
2. Given the set of objects in two consecutive frames, we compute the pairwise IOU of each object in the two frames. We use a cutoff of 0.7 to call an object the same across consecutive frames

FRAMEQL

Data: Training data, held-out data, unseen video, $uerr \leftarrow$ user's requested error rate, $conf \leftarrow$ user's confidence level
Result: Estimate of requested quantity
train specialized NN on training data;
$err \leftarrow$ specialized NN error rate on held-out data;
$\tau \leftarrow$ average of specialized NN over unseen video;
if $P(err < uerr) < conf$ **then**
    return τ;
else
    $\hat{m} \leftarrow$ result of control variates;
    return $\hat{m}$;
end

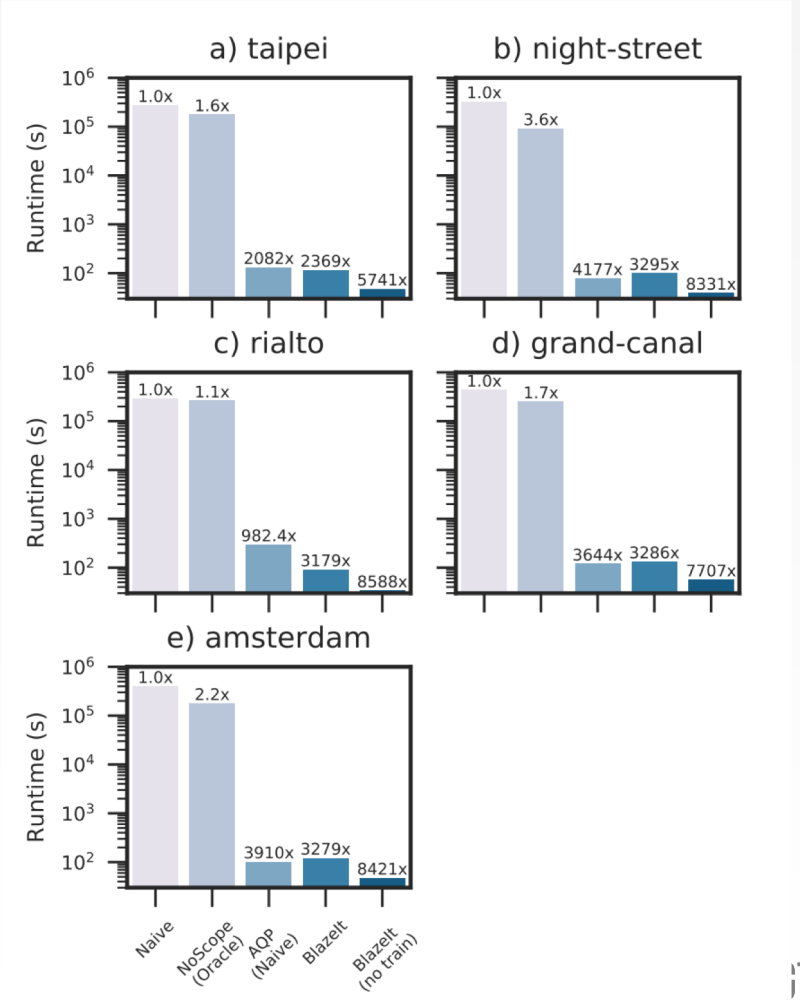
Algorithm 1: BLAZEIT's procedure to return the results of an aggregate query. This is performed when there are enough examples to train a specialized NN.

EVALUATION

1. Aggregate queries
2. Scrubbing queries for rare events
3. Accurate, spatiotemporal queries over a variety of object classes

1. 4000× increased throughput compared to a naive baseline, a 2500× speedup compared to NOSCOPE, and up to a 8.7× speedup over AQP
2. 1000× speedup compared to a naive baseline and a 500× speedup compared to NOSCOPE for **video scrubbing queries**
3. 50× speedup for **content-based selection** over naive methods by automatically inferring filters to apply before object detection

AGGREGATE QUERIES



- Naive: object detection on every frame.
- NOSCOPE oracle: the object detection method on every frame with the object class present.
- Naive AQP: sample from the video.
- BLAZEIT: use specialized NNs and control variates for efficient sampling.
- BLAZEIT (no train): exclude the training time from BLAZEIT.

SCRUBBING QUERIES

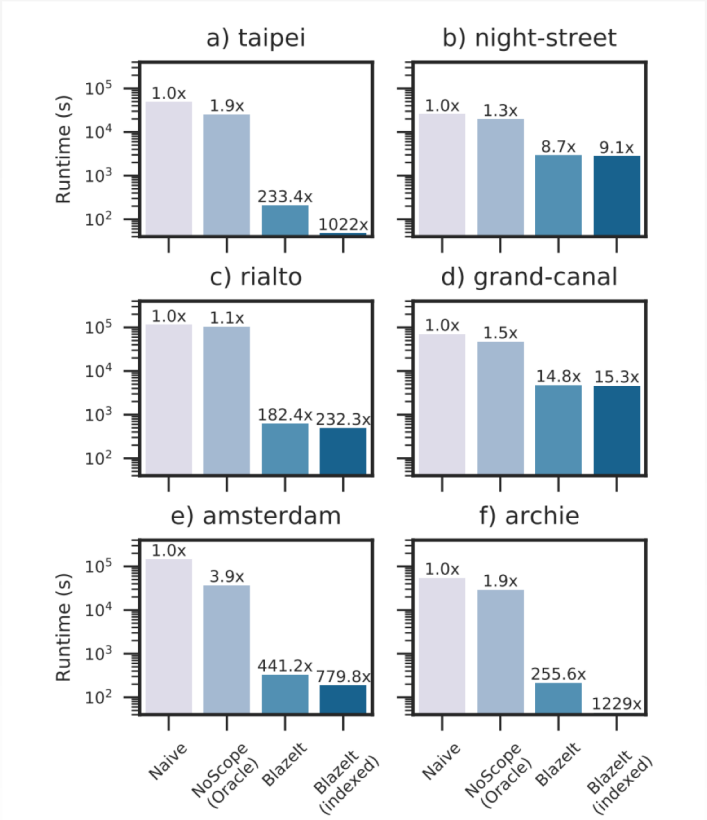


Figure 6: End-to-end runtime of baselines and BLAZEIT on scrubbing queries. Note the y-axis is on a log-scale. All queries looked for 10 events. The average over three runs is shown.

- Naive: the object detection method is run until the requested number of frames is found.
- NOSCOPE: the object detection method is run over the frames containing the object classes of interest until the requested number of frames is found.
- BLAZEIT: specialized NNs are used as a proxy signal to rank the frames
- BLAZEIT (indexed): assume the specialized NN has been trained and run over the remaining data, as might happen if a user runs queries about some class repeatedly.

CONTENT-BASED SELECTION QUERIES

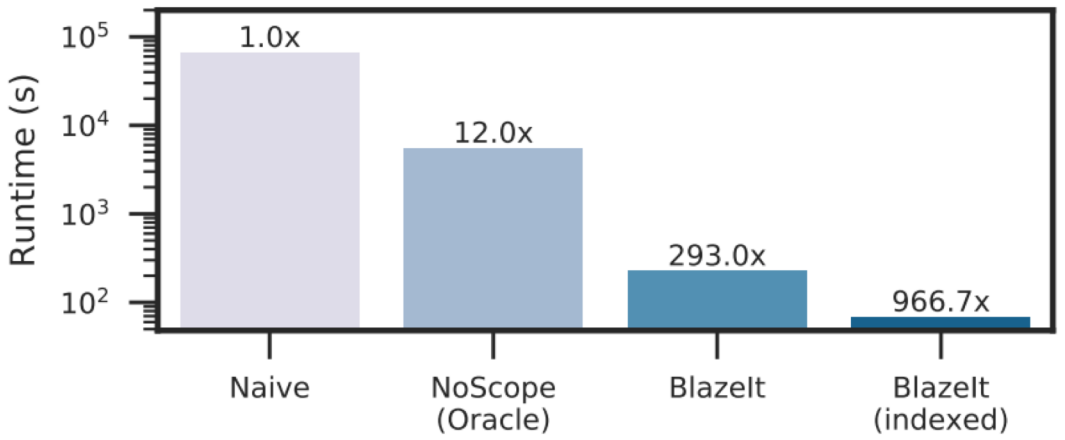


Figure 8: End-to-end runtime of baselines and BLAZEIT on finding at least one bus and at least five cars in `taipei`. Note the y-axis is on a log scale.

- Naive: run the object detection method on every frame.
- NOSCOPE oracle: run the object detection method on the frames that contain the object class of interest.
- BLAZEIT:

CONCLUSION

- Querying video for semantic information has become possible with recent advances in computer vision, but these models run as much as 10× slower than real-time.
- FRAMEQL, and BLAZEIT, a system that accepts, automatically optimizes, and executes FRAMEQL queries up to three orders of magnitude faster
- FRAMEQL can answer a range of real-world queries, of which we focus on exploratory queries in the form of aggregates and searching for rare events

NEW IDEAS IN THIS PAPER

- Introduced new algorithms using deep learning (specialized NN in importance sampling for finding rare events)
- Specialized SQL language can be greatly helpful for domain specific tasks:
  - FRAMEQL, a query language for spatiotemporal information of objects in videos

NEXT RESEARCH DIRECTIONS

- Adding Unsupervised/limited label(semi-supervised) deep learning algorithms
- Solving Limitations of BlazeIt
  - Model Drift: different distribution of the datasets
  - Labeled set: Warm starting of the filters
  - Object detection: user defined object detection classes