

THIA: Accelerating Video Analytics using Early Inference and Fine-Grained Query Planning

Jiashen Cao

Georgia Institute of Technology

Joy Arulraj

Georgia Institute of Technology

Ramyad Hadidi

Georgia Institute of Technology

Hyesoon Kim

Georgia Institute of Technology

Abstract

To efficiently process visual data at scale, researchers have proposed two techniques for lowering the computational overhead associated with the underlying deep learning models. The first approach consists of leveraging a specialized, lightweight model to directly answer the query. The second approach focuses on filtering irrelevant frames using a lightweight model and processing the filtered frames using a heavyweight model. These techniques suffer from two limitations. With the first approach, the specialized model is unable to provide accurate results for hard-to-detect events. With the second approach, the system is unable to accelerate queries focusing on frequently occurring events as the filter is unable to eliminate a significant fraction of frames in the video.

In this paper, we present THIA, a video analytics system for tackling these limitations. The design of THIA is centered around three techniques. First, instead of using a cascade of models, it uses a single object detection model with multiple exit points for short-circuiting the inference. This early inference technique allows it to support a range of throughput-accuracy tradeoffs. Second, it adopts a fine-grained approach to planning, and processes different chunks of the video using different exit points to meet the user’s requirements. Lastly, it uses a lightweight technique for directly estimating the exit point for a chunk to lower the optimization time. We empirically show that these techniques enable THIA to outperform two state-of-the-art video analytics systems by up to 6.5 \times , while providing accurate results even on queries focusing on hard-to-detect events.

1 Introduction

Researchers have proposed systems for quickly processing visual data with a tolerable drop in accuracy [3, 4, 7, 8, 10–13, 16, 23, 24]. These systems detect objects in videos using deep neural networks (DNNs) [5, 17]. The key challenge that these systems tackle is the computational overhead of the underlying object detection model.

PRIOR WORK. To efficiently process visual data at scale, researchers have proposed two techniques. The first approach, presented in BLAZEIT [10], consists of leveraging a specialized, lightweight model to directly answer the query. The second approach, introduced in PP [13], focuses on filtering irrelevant frames using a lightweight model. The frames that pass through the filtering model are then processed by the heavyweight object detection model (illustrated in Figure 2). So, these systems accelerate query processing by *not* processing a subset of video frames using the heavyweight model. However, these techniques suffer from two limitations. With the first approach, the specialized model is unable to provide accurate

results for hard-to-detect events. With the second approach, the system is unable to accelerate queries focusing on frequently occurring events. This is because the filter is unable to eliminate a significant fraction of frames in the video.

Another line of research, illustrated in TAHOMA [1], focuses on leveraging a collection of differently sized models to process the frames based on the complexity of the event. However, using such a cascade of models comes with two limitations. First, switching from one model to another in the GPU is expensive. This switching overhead is further exacerbated if we seek to frequently change the model to process different subsets (*i.e.*, chunks) of the video to maximize performance. For instance, loading a Faster-RCNN model in PyTorch on an NVIDIA Titan Xp GPU takes 2 s (including framework initialization and model loading). Second, using a collection of models to support different throughput-accuracy tradeoffs does not scale well due to the large GPU memory footprint of these models.

Prior efforts have mostly focused on altering the design of the inference pipeline. However, they do not elaborate on how to adapt this pipeline (*e.g.*, when to use a particular model) based on the chunk. They choose a single plan for the entire video based on the profiling results obtained on a set of sampled frames. Such a coarse-grained approach to query planning does not leverage the variation in the frequency and detection difficulty of different events in a video. If objects are difficult to detect, this approach leads to less accurate results. On the other hand, if objects are easier to detect, a conservative coarse-grained query plan significantly increases the query processing time (but returns correct results). We defer a detailed discussion of these limitations to §3.

OUR APPROACH. In this paper, we present THIA, a video analytics system for tackling the limitations highlighted above. THIA leverages three techniques to accelerate queries over visual data.

First, it uses a *single* object detection model with multiple points for short-circuiting the inference. These *exit points* (EPs) offer a set of throughput-accuracy tradeoffs. While processing the query, THIA uses a shallow EP to quickly process frames that are irrelevant or contain easy-to-detect events. If the frames contain hard-to-detect events, then THIA falls back to a deeper EP in the model to deliver higher accuracy. This EARLY INFERENCE technique eliminates the switching overhead and lowers the GPU memory footprint of THIA.

Second, THIA adopts a fine-grained approach to planning. It processes different chunks of the video using different EPs to meet both the performance and accuracy requirements (elaborated in §6.1). This FINE-GRAINED PLANNING technique increases the optimization time of a query. To lower this overhead, we present a third technique to quickly decide which EP to use for a given chunk. THIA

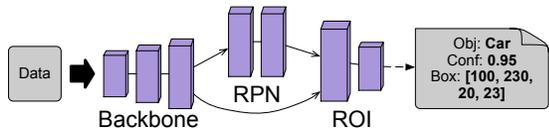


Figure 1: Object Detection Model – Components of the model.

uses a shallow model for EXIT POINT ESTIMATION instead of running inference on the sampled frames. We evaluate a set of queries focusing on events with different levels of frequency and detection difficulty on two traffic surveillance datasets: UA-DeTrac [21] and Jackson Town [10]. On all of the queries, THIA outperforms the state-of-the-art video analytics systems by up to 6.5× with a tolerable drop in accuracy.

CONTRIBUTIONS. Our research makes the following contributions:

- We present the EARLY INFERENCE technique to construct a single model that offers a set of throughput-accuracy tradeoffs for challenging vision tasks like object detection.
- We propose a FINE-GRAINED PLANNING technique that works in tandem with the EARLY INFERENCE technique.
- We present the EXIT POINT ESTIMATION technique to reduce the optimization overhead of the FINE-GRAINED PLANNING technique.
- We implement all of these techniques in THIA and show that it outperforms two state-of-the-art video analytics systems on a wide range of queries.

2 Background

We now present an overview of object detection and sampling techniques used in video analytics systems (§2.1 and §2.2). We later discuss the key techniques used in state-of-the-art systems (§2.3).

2.1 Object Detection

Object detection models usually contain three components: (1) backbone network, (2) region proposal network (RPN), and (3) region of interests network (ROI), as illustrated in Figure 1. The backbone network extracts the high-level features from a frame. Then, the RPN and ROI networks determine the location and type of objects detected in the frame. Data flows from the backbone network to RPN, and ROI returns the final prediction results (object category, location within the frame, and confidence score).

In machine learning literature, the *oracle* model returns the correct answer to all queries. However, in practice, there is no ground truth for unseen data. Similar to prior efforts, we assume that the most accurate model, which also tends to be the most compute-intensive model, is the oracle model [9–11, 13, 24].

2.2 Sampling

Sampling is a frequently used technique for processing visual data at scale. By processing only a subset of frames using the object detection model, a video analytics system lowers the overall query processing time. For example, BLAZEIT [10] uses uniformly random sampling to process aggregate queries (e.g., counting the average number of cars within a given period of time).

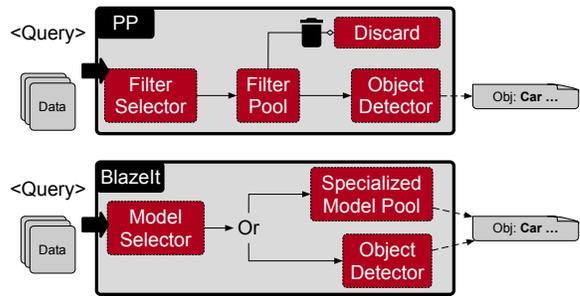


Figure 2: Architecture of Video Analytics Systems – Architecture of two state-of-the-art video analytics systems: (1) PP [13] and (2) BLAZEIT [10].

	+ MS	+ MC	+ FP †	+ EI †	+ EP-Est †
NAIVE					
PP	✓				
BLAZEIT	✓				
THIA-SINGLE *			✓		
THIA-MULTI *		✓	✓		
THIA-EI *			✓	✓	
THIA *			✓	✓	✓

MS: Model Specialization, MC: Model Cascade, FP: FINE-GRAINED PLANNING, EI: EARLY INFERENCE, EP-Est: EXIT POINT ESTIMATION.
 †: Techniques used in THIA. ★: Variants of THIA.

Table 1: Qualitative Comparison of Video Analytics Systems – Key characteristics of state-of-the-art video analytics systems.

THIA uses sampling for a different purpose (elaborated in §6). A chunk is a continuous segment of frames within a video. THIA’s OPTIMIZER constructs plans at chunk-level granularity (instead of video-level granularity) to lower the query processing time. Query processing time consists of two components: (1) optimization time, and (2) execution time. During the optimization phase, THIA generates a plan for each chunk. During the execution phase, it runs these plans.

2.3 State-of-the-Art Systems

Table 1 lists the key characteristics of several state-of-the-art video analytics systems: (1) PP [13], (2) BLAZEIT [10] (3) Miris [2], (4) Tahoma [1], and (5) Panorama [24]. We present the benefits and limitations of the first two systems in §1. Their architectures are illustrated in Figure 2.

Miris [2] is a video analytics system that focuses on multi-object tracking. It uses coarse-grained sampling to gain a high-level perspective of the video and then gradually increases the sampling rate to improve the accuracy of tracking. THIA differs from Miris in two ways. First, it is tailored for object detection. Second, it only samples for query planning (not for query execution). In §7.7, we illustrate the benefits of other optimizations in THIA by comparing it against a variant of THIA that only uses FINE-GRAINED PLANNING (THIA-SINGLE in Table 1).

TAHOMA [1] is another closely related analytics system. It constructs a model cascade by combining a chain of image classification models and determines when to short-circuit the inference based on

Query	SQL	Predicate Frequency	Predicate Difficulty
Q1	Select frameID From UA-DeTrac Where Count (Car) >= 4;	Frequent	Easy
Q2	Select frameID From UA-DeTrac Where Count (Truck) >= 1;	Frequent	Hard
Q3	Select frameID From UA-DeTrac Where Count (Bus) >= 4;	Rare	Hard
Q4	Select frameID From Jackson-Town Where Count (Car) >= 4;	Rare	Hard

Table 2: List of Queries – Queries with varying frequency and levels of difficulty in detecting events.

the confidence score of prediction of each model. Unlike TAHOMA, THIA is geared toward object detection. So, the inference result consists of a set of confidence scores for all the objects present in the frame. It is challenging to short-circuit the inference pipeline based on a set of confidence scores. In THIA, the EARLY INFERENCE technique is guided by query accuracy (not model accuracy). In §7.7 and §7.8, we illustrate the limitations of using a model cascade by comparing THIA against a variant of THIA that uses FINE-GRAINED PLANNING along with a model cascade (THIA-MULTI in Table 1), instead of a single EARLY INFERENCE model¹.

Panorama [24] is another state-of-the-art video video analytics system that uses a single model to solve the unbounded vocabulary problem in object recognition. While this system also offers a set of throughput-accuracy tradeoffs similar to THIA, it is geared towards comparing embeddings from two input frames. So, it selects the EP based on the delta between two embeddings while extracting the embeddings. Lastly, it clusters these embeddings to recognize the objects in the input frames. In contrast, THIA uses EARLY INFERENCE in the object detection model itself and seeks to reduce optimization time using the EXIT POINT ESTIMATION technique.

3 Motivation

In this section, we discuss the limitations of PP and BLAZEIT to motivate the need for THIA. We focus on the four queries described in Table 2. These queries differ in: (1) frequency of appearance of target objects in the video, and (2) level of difficulty in providing a correct answer to a query.

LIMITATION I – MODEL SPECIALIZATION OVERHEAD. Both PP and BLAZEIT rely on specialized models. PP uses a specialized model as a filter. Since each filter detects only one object category, it needs to train multiple lightweight models (*i.e.*, filters) during runtime to support different object categories. BLAZEIT uses a specialized model to directly return the results. A model may directly return the count of cars in an image, so it must maintain multiple models for different predicates (*e.g.*, **Count**(Car) is a predicate). With this

¹THIA-MULTI delivers better performance than a naive model cascade due to the FINE-GRAINED PLANNING technique. With a naive model cascade, the system cannot directly process frames with the optimal model. It must use all the smaller models before stopping the inference at the optimal model.

Difficulty	Precision	Recall	Throughput
Easy	97.72%	67.98%	12.98×
Hard	100.00%	100.00%	0.93×

Table 3: BLAZEIT vs NAIVE – Key metrics of BLAZEIT with respect to a NAIVE system that only uses the heavyweight object detector.

model specialization technique, these systems need to train and maintain models for different objects and predicates, respectively. We seek to reduce model maintenance overhead by offering a range of accuracy and query execution time tradeoffs in a single model.

LIMITATION II – FREQUENT EVENTS. The filtering technique used in the PP system [13] relies on data reduction by the filter to achieve speedup. Let’s assume the system is processing N frames and that the fraction of frames that is filtered and discarded by the filter is r . Let the costs of running the filter and running the object detector be C_f and C_o per frame, respectively. To obtain a speedup, the data reduction rate must satisfy this constraint:

$$N(C_f + (1 - r) \cdot C_o) < NC_o \equiv r > \frac{C_f}{C_o}$$

This constraint is not met by frequent events (*e.g.*, Q1 in Section 3). In this case, since r is small, the filter slows down the overall pipeline since it adds additional overhead. As a result, PP is slower than NAIVE (*i.e.*, naively running object detector on every frame) for frequent queries like Q1. PP only provides a 0.93× speedup compared to NAIVE in this case. Instead, for rare queries like Q3, PP is able to provide a 1.44× speedup compared to NAIVE. We seek to dynamically adjust the query execution pipeline based on the estimated frequency of the event.

LIMITATION III – DIFFICULT-TO-DETECT OBJECTS. BLAZEIT [10] uses a specialized model to directly return aggregates (*e.g.*, number of cars in an image). This approach does not generalize to complex visual datasets. The reasons are twofold. First, the specialized model is designed to be shallow for fast execution. So, it is unable to learn complex patterns. Second, it relies on an ad-hoc subset of videos for training, so the lack of positive examples greatly affects the quality of the model.

As shown in Table 3, BLAZEIT returns precise answers for easy-to-answer queries. However, it has a lower recall metric. For hard-to-answer queries (*e.g.*, Q3), the specialized model does not offer useful results. So, the system instead falls back to the object detection model. In this case, BLAZEIT runs the specialized model, resulting in lower performance than NAIVE. In contrast, THIA is capable of selecting an optimal plan with good accuracy and performance metrics.

OUR APPROACH. In Figure 3, we show two prediction results from our EARLY INFERENCE technique (the oracle object detection EP and a shallow EP, respectively). We observe that the faster EP is still able to capture the presence of cars, but it is less accurate in two ways. First, the bounding boxes are not accurate, so multiple bounding boxes are returned for the same object. Second, it tends to miss hard-to-detect objects (*e.g.*, objects far away or objects with lights). If a user queries for an image with exactly four cars, THIA uses the oracle exit point to satisfy the precision requirement. However,



(a) Results of oracle EP- 1x speedup. (b) Results of shallow EP- 6x speedup.

Figure 3: Objects Detection Results – Objects detected by (a) oracle, and (b) shallow EP.

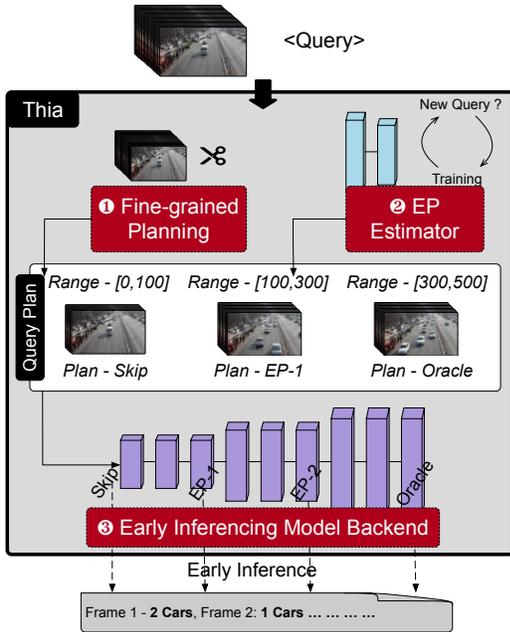


Figure 4: System Overview – The two major components of THIA are: (1) OPTIMIZER and (2) EXECUTION ENGINE. While the OPTIMIZER relies on FINE-GRAINED PLANNING and EXIT POINT ESTIMATION techniques, the EXECUTION ENGINE performs EARLY INFERENCE.

if the user is only interested in images with cars, THIA uses the faster exit point to obtain a 6x speedup. We design THIA so that it carefully chooses the optimal query execution plan for every chunk of the video to deliver higher accuracy and speedup.

4 System Overview

Figure 4 illustrates the architecture of THIA.

1 FINE-GRAINED PLANNING. When the system gets a query, the OPTIMIZER uses the FINE-GRAINED PLANNING technique to construct a query execution plan. It first splits the entire video into a set of small chunks. The size of a chunk is determined dynamically at runtime (covered in §6). For each chunk, the OPTIMIZER chooses the optimal plan (*i.e.*, when to stop inference in the model). Such a fine-grained query plan enables THIA to deliver higher accuracy and throughput compared to a coarse-grained plan for the entire video. A naive technique for picking the plan consists of running the model on a set of sampled frames from the chunk. While the

fine-grained plan reduces the query execution time (THIA-EI in Table 1), it increases the query optimization time, which hurts the overall query processing time (discussed in §7.5). To reduce the optimization time, THIA instead leverages a more lightweight EXIT POINT ESTIMATION technique.

2 EXIT POINT ESTIMATION. THIA uses EXIT POINT ESTIMATION and FINE-GRAINED PLANNING techniques in tandem to reduce the overhead of the OPTIMIZER. The OPTIMIZER uses a shallow neural network to directly estimate when to short-circuit the inference in an EARLY INFERENCE model. It trains an EP estimator for every unique query executed in the system. We discuss how THIA obtains data for training the EXIT POINT ESTIMATION model in §6.

3 EARLY INFERENCE. The fine-grained query plan constructed by the OPTIMIZER consists of a list of chunks and the model chosen for each chunk. For example, THIA may skip frames 0 through 100, run EP-1 on frames 101 through 300, and evaluate the oracle EP (*i.e.*, EP-3) on frames 300 through 500. The EXECUTION ENGINE takes this query plan and uses the EARLY INFERENCE technique to deliver different accuracy-performance tradeoffs with a single model.

5 EARLY INFERENCE

In this section, we present the EARLY INFERENCE technique. We first provide an overview of this technique in §5.1. We then illustrate its utility using a case study with Faster-RCNN [17] in §5.2.

5.1 Overview

We seek to construct a *single* model with multiple exit points wherein the inference may be short-circuited to improve performance at the expense of accuracy. We do *not* want to construct a collection of models to accomplish this goal. The OPTIMIZER dynamically adjusts the EP based on the query. If the query is relatively easy to answer, THIA delivers higher speedup by stopping the inference earlier (while returning accurate results). We discuss how THIA estimates the correct EP for a chunk in §6. In this section, we focus on how we construct a model with multiple EPs.

As discussed in §2.1, object detection models usually rely on a backbone network that is based on a state-of-the-art image classification model (*e.g.*, ResNet-50 [6] and VGG-16 [18]). Since these classification models are tailored for high accuracy, they consist of a stack of compute-intensive layers that lead to lower inference throughput. The layers in a backbone network are sequentially connected to each other. Our key idea is to provide faster detection results with lower accuracy by using the features from earlier layers in the backbone network.

MODEL CASCADING VS EARLY INFERENCE: Researchers have proposed model cascades for face recognition [19, 20]. Similar to EARLY INFERENCE, in a model cascade, the features from earlier layers in the backbone network are used for face recognition. However, these techniques differ in two ways. First, face recognition is a binary classification task (*i.e.*, face exists in the image or not). So, the additional classification layers are only instrumented in this approach. Second, these efforts propose a bespoke architecture to construct the cascade. We instead seek to support early inference in widely used object detection models.

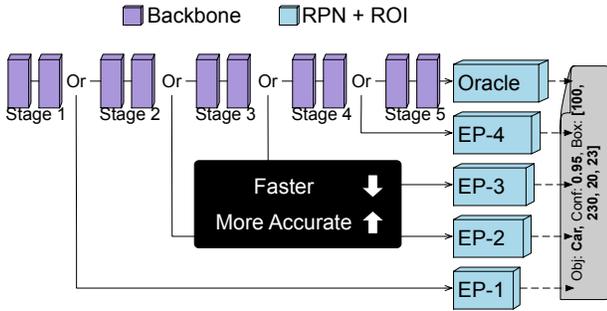


Figure 5: EARLY INFERENCE in Faster-RCNN – Architecture of a Faster-RCNN model that supports early inference.

To support EARLY INFERENCE with a general-purpose object detection model, we introduce additional RPN and ROI units in the earlier layers of the backbone network. We modify the number of parameters in these units so that they operate on the feature tensor emitted by the backbone network. As shown in Figure 5, for a given input, OPTIMIZER may choose to short-circuit the inference using the newly added units to speed up inference.

5.2 Case Study: Faster-RCNN

Faster-RCNN is a state-of-the-art object detector [17]. We now discuss how we extend this model to support EARLY INFERENCE. We next describe how to generalize the training process to other models.

FASTER-RCNN WITH EARLY INFERENCE: The backbone network of Faster-RCNN is the ResNet-50 [6] model. ResNet-50 consists of five stacked compute blocks, so we extend this model to support five EPs (we could support fewer or additional EPs if needed by instrumenting other layers of the backbone network). The default output of the model corresponds to the fifth EP. We add four additional EPs that provide a wide set of throughput-accuracy tradeoffs (*i.e.*, EP-1, EP-2, EP-3, and EP-4 in Figure 5). We refer to EP-5 as the oracle (since it is the output of the original model). We preserve the structure of RPN and ROI units as is the case of the oracle. However, we modify the first layer in these units to work with the output tensors of the early EPs that vary in size. Table 4 lists the layer configuration of each EP ².

TOP-DOWN TRAINING: We adopt a novel top-down training technique for constructing models that support early inference. We start the training process with the following multi-loss function:

$$L(x; Y) = \frac{1}{|E|} \sum_{e \in E} L(\{y_c\}, \{y_t\}; Y)$$

E represents the set of exit points (including the oracle). L denotes the object detection loss function used in Faster-RCNN [17]. This training step tunes all EPs.

E represents all possible object detection EPs, including the oracle EP. We begin with the oracle EP. The reasons for doing this are twofold. First, the oracle gets the features emitted by the last stage of the backbone network, so training this EP ensures that all layers

²We found that upsampling the input channel size to 2048 does not improve accuracy since the features from earlier EPs are coarse.

Exit Points	Layer Config		Performance		
	Channel	Kernel	Speedup	TPr	FNr
EP-1	64	3 × 3	6.90×	87.99%	42.70%
EP-2	256	3 × 3	2.62×	91.65%	26.95%
EP-3	512	3 × 3	2.46×	95.52%	16.22%
EP-4	1024	3 × 3	1.97×	98.17%	6.56%
EP-5 (Oracle)	2048	3 × 3	1.00×	100.00%	0.00%

TPr: True positive ratio. FNr: False negative ratio.

Table 4: EARLY INFERENCE model knobs – the layer configuration and performance of each object detection knob in EARLY INFERENCE model.

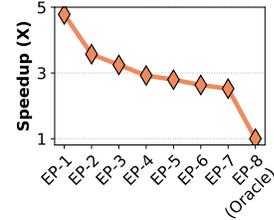


Figure 6: Generalization of EARLY INFERENCE – Application of the EARLY INFERENCE technique to a VGG-16 model for image classification.

converge to the optimal state. Second, we seek to ensure that the oracle EP in the EARLY INFERENCE model delivers the same accuracy as that of the original model. After training the oracle EP, we freeze all the layer parameters in the backbone network. This ensures that fine-tuning the shallow EPs later does not affect the previously tuned EPs. We gradually fine-tune the RPN and ROI units starting from EP-4 through EP-1.

THROUGHPUT-ACCURACY TRADEOFFS. Table 4 lists the speedup of shallow EPs with respect to the the oracle EP. For a given video frame, this EARLY INFERENCE model offers up to a 6.9× speedup when we stop the inference at the first EP. Table 4 also summarizes the true positive and false negative percentage of each EP on the training dataset with respect to the oracle EP. These metrics are averaged across all categories. Shallower EPs return more false negatives and fail to return a few true positives. In other words, they are more likely to not return a positive frame instead of misclassifying a negative frame. If the system were to use a shallow EP for the entire video or sequence of images, the impact on query accuracy would be significant. Instead, it must use the oracle EP on some difficult chunks of the video. We cover this FINE-GRAINED PLANNING technique in §6. In §7, we demonstrate that THIA has a tolerable accuracy loss using both EARLY INFERENCE and FINE-GRAINED PLANNING techniques.

GENERALIZATION. The EARLY INFERENCE technique generalizes to other models (*e.g.*, VGG-16 [18]) and other vision tasks (*e.g.*, image classification). This is because most of these deep learning models contain similar backbone networks that benefit from the EARLY INFERENCE technique. Furthermore, the number of EPs may be increased or decreased based on the complexity of the model.

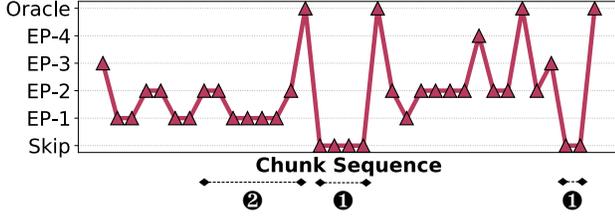


Figure 7: Variation of Optimal EP – The fastest, accurate EP in an EARLY INFERENCE model for a sequence of chunks in a video.

Figure 6 illustrates another EARLY INFERENCE model based on VGG-16 for an image classification task that is trained on the Flower-102 dataset [14]. Here, EP-1 provides a 4.7 \times speedup compared to EP-8 (*i.e.*, the oracle EP). By using all the eight EPs together, the system achieves a 2.7 \times speedup compared to the oracle EP with minimal accuracy loss.

6 Query Planning

We present the FINE-GRAINED PLANNING technique in this section. In §6.1, we make the case for FINE-GRAINED PLANNING. In §6.2, we discuss how THIA samples frames and constructs chunks to apply this technique. Lastly, in §6.3, we introduce the EXIT POINT ESTIMATION technique for reducing the optimization time.

6.1 Motivation

As we discussed in §5.2, the EARLY INFERENCE model contains a set of EPs. The goal of the OPTIMIZER is to choose an optimal (accurate and fast) EP for every fine-grained chunk of the video at runtime. Our key observation is that the optimal EP changes at chunk granularity. Figure 7 illustrates the chunk-level query plan for Q4 in Section 3. The triangles in Figure 7 represent the fastest (but still accurate enough) EP for every chunk in the video. This example shows that the optimal EP constantly changes. So, it is essential to dynamically adjust the query plan at *runtime* to achieve both good accuracy and performance.

State-of-the-art systems (*e.g.*, BLAZEIT [10] and PP [13]) take a coarse-grained approach to planning. They choose a single plan for the entire video based on the accuracy of the model on a set of sampled frames. The limitations of this technique are twofold.

PERFORMANCE DEGRADATION. Positive events tend to not appear in every chunk of the video (*i.e.*, selectivity of the predicate is typically high). If we pick a static plan for the entire video, video chunks that are less likely to contain positive events or that contain easy-to-detect events are passed to a more compute-intensive EP. Thus, the system does not leverage the opportunity to further improve performance by either skipping those chunks or using less compute-intensive EPs for those chunks. With FINE-GRAINED PLANNING, THIA uses a faster EP or directly skips the entire chunk (① in Figure 7).

ACCURACY LOSS. The distribution of the target event and the accuracy of the model vary across the video. A statically selected, shallow EP will hurt accuracy by missing hard-to-detect events. As shown in Figure 7, some chunks require deeper EPs to make

Algorithm 1: Fine-grained query planning.

```

Input :  $V$  - Video data.
          $EP\text{-List}$  - The list of EPs in the EARLY INFERENCE model.
          $P$  - Precision constraint of the query.
          $R$  - Recall constraint of the query.
Output: Return a list of fine-grained plans.
1   $video\_length \leftarrow Length(V)$ 
   // estimate initial sampling rate.
2   $sampling\_rate \leftarrow EstimateSamplingRate(V)$ 
   // optimize the query plan.
3  return  $GetQueryPlan(V, EP\text{-List}, P, R, sampling\_rate)$ 

4  Function  $PickBestEP(V\_sub, EP\text{-List}, P, R)$ 
   Output: Return the optimal EP under  $P$  and  $R$  constraints, and
           the rate of positive frames in the sampled subset.
5  Function  $GetQueryPlan(V, EP\text{-List}, P, R, sampling\_rate)$ 
   Output: A collection of fine-grained plans.
   // divide into smaller chunks.
6   $sampling\_span \leftarrow \frac{1}{sampling\_rate}$ 
7   $V\_sub \leftarrow V[0, 1 \cdot sampling\_span, 2 \cdot sampling\_span \dots]$ 
8   $best\_ep, posi\_ratio \leftarrow PickBestEP(V\_sub, EP\text{-List}, P, R)$ 
   // the collection of fine-grained plans.
9   $plan \leftarrow \{\}$  if ( $posi\_ratio$  is sufficient and  $best\_ep$  is fastest) or
   ( $Length(V)$  is small) then
10 |    $plan \ += \{V: best\_ep\}$ 
11 else if  $posi\_ratio$  is insufficient then
12 |    $plan \ += \{V: skip\}$ 
13 else
14 |   for  $V\_chunk \in V$  do
15 |       // double the sampling rate.
16 |        $plan \ += GetQueryPlan(V\_chunk, EP\text{-List}, P, R,$ 
    $2 \cdot sampling\_rate)$ 
   return  $plan$ 

```

accurate predictions (② in Figure 7). With FINE-GRAINED PLANNING, THIA dynamically adjusts the plan based on the difficulty of detecting the target event.

6.2 Chunking Algorithm

When THIA gets a query, it first splits the given video into a set of chunks. It then samples a set of frames from each chunk and then evaluates the accuracy of all the EPs in the EARLY INFERENCE model on these sampled frames. Using these results, the system selects the best EP for each chunk. Lastly, it executes the query using the selected plan. The key components of the algorithm that THIA uses for chunking videos are as follows:

① **HIERARCHICAL CHUNKING.** The two key decisions made by the OPTIMIZER are: (1) chunk size, and (2) sampling rate (*i.e.*, the number of frames to pick from a chunk). The system delivers higher accuracy with a higher sampling rate since more samples allow it to better estimate the optimal EP for each chunk. However, this hurts throughput since the system must evaluate the model’s behavior on more frames, thereby increasing optimization time. Choosing the chunk size is also a challenging task. This is because the duration

of an event varies based on the video, so THIA must dynamically adjust the chunk size at runtime.

To tackle these challenges, THIA takes a hierarchical approach for picking the chunk size and the sampling rate for each chunk. It initially uses a large chunk size and a low sampling rate. This allows the OPTIMIZER to gain a rough understanding of the contents of the video based on the inference results collected using the sampled frames. Based on this knowledge, it recursively adjusts the chunk size and sampling rate.

Algorithm 1 presents the hierarchical, recursive technique used by the OPTIMIZER. As shown in Line 9, the recursive algorithm stops when the chunk size is smaller than a threshold or if the fastest EP has been chosen for a given chunk that contains enough positive frames. In Line 8, the PickBestEP function returns the rate of positive frames in a chunk (*i.e.*, `posi_ratio`) that is obtained from the oracle EP. It is important to ensure that the chunk has sufficient positive frames, since the calculated precision and recall metrics of the EPs do not generalize well without sufficient positive frames. These constraints bound the optimization time. As shown in Line 11, if there are very few positive frames, then THIA skips the entire chunk to reduce both optimization time and execution time. Lastly, it gradually reduces the chunk size and increases the sampling rate, as shown in Line 15. The intuition is that if the system is not able to select a plan based on its coarse-grained understanding of the chunk, it must sample more frames from that chunk in the next iteration. By using a small chunk size, the OPTIMIZER is able to adjust the plans quickly to transient events.

② **SAMPLING RATE BOUNDS.** The OPTIMIZER gradually increases the sampling rate to improve the quality of its plan. However, this increases the optimization time and may hurt the overall throughput obtained with the plan. This is because the decrease in the query execution time is not sufficient to justify the increase in the optimization time. To overcome this limitation, the OPTIMIZER uses the following constraint to bound the initial sampling rate (Line 2):

$$sampling_rate * 2^{\lceil \log \frac{|V|}{100} \rceil} \leq 0.1$$

Here, we assume that chunks must contain at least 100 frames and we seek to bound the final sampling rate to 0.1 even in the worst-case setting. The maximum depth of the recursive algorithm is $\lceil \log \frac{|V|}{100} \rceil$ (sampling rate is doubled in each iteration).

③ **MEMOIZATION OF INFERENCE RESULTS.** In Line 7, the newly picked samples could be different from those that have already been evaluated. Evaluating all the EPs on a sample is expensive. To reduce this overhead, the OPTIMIZER memoizes the inference results and reuses the results of *nearby* frames. This technique is illustrated in Figure 8. Without memoization, the results for the second and fourth frames must be obtained again when the sampling rate is increased in the next iteration. THIA instead reuses the results of nearby frames within the same chunk. With memoization, it picks the cached results for the third frame instead of running inference on the fourth frame. Thus, it evaluates only the EPs on the second frame.

④ **EVALUATION-BASED EP SELECTION.** To select the best EP, as shown in Line 8, THIA evaluates all the EPs on the sampled frames and compares them with the oracle EP. It picks the fastest EP that

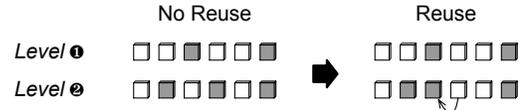


Figure 8: Sampling results no reuse vs reuse – an illustrative example about sampling results about performance saving with no reuse and reuse.

provides 0.8 precision and 0.8 recall. These constraints empirically offer maximal speedup with minimal accuracy loss (§7.2). Even with all of these optimizations, evaluating the EPs on a frame comes with non-trivial optimization time. We next present the EXIT POINT ESTIMATION technique for further reducing the optimization time.

6.3 EXIT POINT ESTIMATION

We seek to reduce the optimization time associated with the EARLY INFERENCE technique. As we present in §7.5, it is important to balance the tradeoff between optimization time and execution time to improve the overall query processing time.

The EXIT POINT ESTIMATION technique consists of using a shallow, two-layer neural network instead of the evaluation step in query planning. The neural network directly returns the optimal EP based on the backbone features. This allows the OPTIMIZER to eliminate compute-intensive evaluation of all EPs. For example, with Faster-RCNN, the inputs to the EXIT POINT ESTIMATION model are the features emitted by the fifth stage of the EARLY INFERENCE model.

To train this neural network, OPTIMIZER uses 200 images from the training dataset of the EARLY INFERENCE model along with the associated EP decision. For robust results, THIA must train a separate EXIT POINT ESTIMATION model for each query. However, the overhead of training this model is tolerable because: (1) it is a one-time overhead for each query; and (2) training time for the EXIT POINT ESTIMATION model is negligible compared to total query processing time due to the simple structure of the model. We defer an empirical analysis of this optimization to §7.6.

ESTIMATION-BASED EP SELECTION. Since the EXIT POINT ESTIMATION model directly estimates the optimal EP for a video frame, it does not return precision and recall metrics for all the EPs. So, the OPTIMIZER extrapolates these metrics based on the estimated EP. We next discuss how this extrapolation is done.

Let us split the set of sampled frames into two subsets: (1) those that contain positive events as reported by the oracle EP (Line 9 in Algorithm 1), and (2) those that do not contain positive events. We define those two subsets as S and \hat{S} , respectively. For a given video frame x , let us denote the EP estimation model that outputs the optimal EP for that frame by $OPT_{EP}(x)$. The OPTIMIZER estimates the number of true positives (TP), false positives (FP), and false

negatives (FN) for any EP k in the EARLY INFERENCE as:

$$\begin{aligned} TP_k &= \sum_{x \in S} g(x), & g(x) &= \begin{cases} 1, & \text{if } k \geq OPT_{EP}(x) \\ 0, & \text{otherwise} \end{cases} \\ FP_k &= \sum_{x \in \bar{S}} g'(x), & g'(x) &= \begin{cases} 1, & \text{if } k < OPT_{EP}(x) \\ 0, & \text{otherwise} \end{cases} \\ FN_k &= \sum_{x \in S} g''(x), & g''(x) &= \begin{cases} 1, & \text{if } k < OPT_{EP}(x) \\ 0, & \text{otherwise} \end{cases} \end{aligned}$$

Our intuition is that a shallow EP is less accurate than a deep EP. So, for a video frame x , the estimated optimal EP (i.e., $OPT_{EP}(x)$) returns correct results. Then, all EPs after the estimated optimal EP (i.e., $k \geq OPT_{EP}(x)$) should also return correct results, and vice versa. Hence, in the case of positive events, a deeper EP k than the estimated optimal EP provides true positive prediction. On the other hand, a shallower EP k than the estimated optimal EP provides false negative prediction. In the case of negative events, shallower EP k than the estimated optimal EP likely results a false positive. With these projected metrics, the OPTIMIZER derives the precision and recall metrics for an EP k as:

$$Precision_k = \frac{TP_k}{TP_k + FP_k}, Recall_k = \frac{TP_k}{TP_k + FN_k}$$

Lastly, the OPTIMIZER picks the fastest EP that meets the precision and recall constraints (e.g., 0.8), as discussed earlier.

7 Experimental Evaluation

We seek to answer the following questions in our evaluation:

- How effective is the EARLY INFERENCE technique in reducing the query processing time (§7.2)?
- How much does each technique contribute to the overall performance (§7.3)?
- How effective is the FINE-GRAINED PLANNING compared to the coarse-grained planning (§7.4)?
- What is the time spent on query planning and execution (§7.5)?
- How effective is the EXIT POINT ESTIMATION technique in reducing the optimization time (§7.6)?
- How effective is THIA compared to other state-of-the-art systems (§7.7)?
- How does the EARLY INFERENCE technique compare against the model cascade technique (§7.8)?

7.1 Experiment Setup

EVALUATED SYSTEMS. Table 1 lists all the video analytics systems that we compare in our analysis (including the variants of THIA). In the NAIVE system, we apply the oracle EP on every frame. We normalize the accuracy metrics of other systems against those of the NAIVE system. We reimplement two other state-of-the-art systems in our framework for comparative analysis: (1) PP [13], and (2) BLAZEIT [10]. In our implementation, the PP system uses ResNet-34 [6] to filter out unrelated frames. The BLAZEIT system uses a specialized model (ResNet-34) to accelerate queries.

To better understand the performance of THIA, we examine three variants of our system: (1) THIA-SINGLE uses only the FINE-GRAINED PLANNING method with the oracle EP. (2) THIA-MULTI

also uses the FINE-GRAINED PLANNING method along with multiple EPs. Specifically, we use Faster-RCNN models with three backbone networks: ResNet-18, ResNet-34, and ResNet-50 [6] as three EPs. (3) THIA-EI is the closest variant of THIA. It uses the FINE-GRAINED PLANNING along with the EARLY INFERENCE technique (but does not use the EXIT POINT ESTIMATION technique).

DATASETS. We evaluate these systems on two datasets: (1) UA-DeTrac [21], and (2) Jackson-Town dataset from [10]. Both datasets are obtained from traffic surveillance cameras. We focus on four vehicle categories in both datasets: Car, Truck, Bus, and Others.

EVALUATION METRICS. Similar to other video analytics systems [1, 9–11, 13], our evaluation normalizes the results with respect to the oracle model (Faster-RCNN model backed by ResNet-50). So, we provide the F-1 score calculated relative to the results of the oracle model. We also report separate precision and recall metrics for each query. This is important since a user might require fine-grained accuracy requirements (e.g., low precision and high recall). We assume that the decoded video is present on disk.

QUERIES. To evaluate these systems, we use the four queries listed in Table 2. Based on the predicate, the frequency of true positive events and the difficulty of detecting those events vary.

SOFTWARE AND HARDWARE. We implement THIA with the Detectron2 [22] framework in PyTorch [15]. We evaluate these systems on a server with 44 CPU cores and 256 GB memory along with one Titan Xp GPU with 12 GB memory.

MODEL TRAINING. As discussed in §5.2, we construct the EARLY INFERENCE model based on Faster-RCNN. We split the UA-DeTrac dataset into two parts: training and validation subsets. We train the EARLY INFERENCE model on the the training subset. We warm up the training process for 1 epoch, and then each EP in EARLY INFERENCE model is trained for 10 epochs in a top-down manner. Since the Jackson-Town dataset does not have ground-truth labels, we directly apply the EARLY INFERENCE model, which is tailored for the UA-DeTrac dataset. For THIA-MULTI, we train three models: Faster-RCNN based on ResNet-18, Faster-RCNN based on ResNet-34, and Faster-RCNN based on ResNet-50. Each model is trained for 10 epochs.

To train the EXIT POINT ESTIMATION model, we use 1000 images from the UA-DeTrac training set. We split these images into training (200 images) and validation subsets. We construct the training so that the distribution of different EPs is balanced. The training data for this model consists of backbone features for those video frames, and the output is the fastest EP that is accurate enough. We quickly train this shallow network for 20 epochs.

7.2 Impact of EARLY INFERENCE

In this experiment, we compare the query processing time of THIA-EI to that of other video analytics systems. The results are shown in Figure 9. The bottom right corner represents the ideal case (faster execution with accurate predictions).

THIA-EI. The most notable observation is that THIA-EI outperforms other systems on most queries. THIA-EI uses both EARLY INFERENCE and FINE-GRAINED PLANNING. On Q1 and Q2, since the fraction of frames filtered out is limited, using an extra specialized model

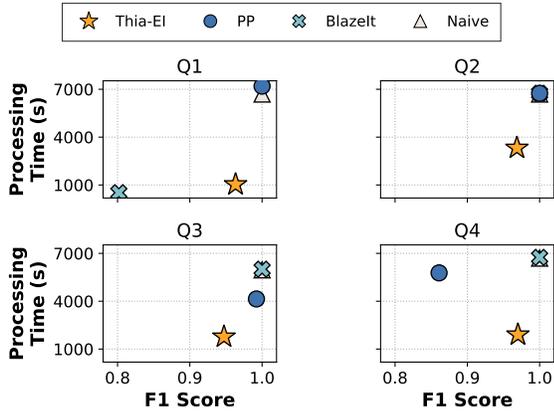


Figure 9: Impact of EARLY INFERENCE – Query processing time and F-1 scores delivered by THIA-EI, PP, and BLAZEIT (bottom right corner represents the ideal system).

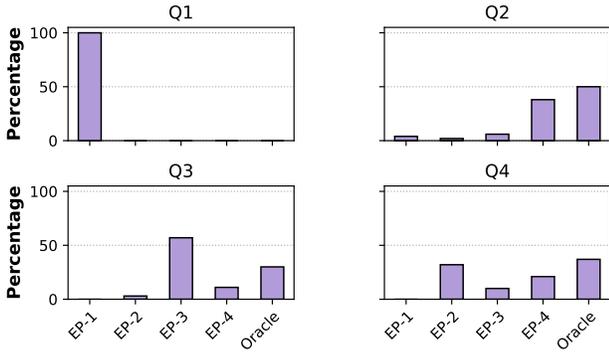


Figure 10: Usage of EPs in EARLY INFERENCE model – Percentage of frames processed using the EPs in the EARLY INFERENCE model.

before the object detector adds additional execution overhead. THIA-EI consistently reduces the total runtime and also delivers a higher F-1 score compared to other systems. In particular, it is 2 – 6× faster than NAIVE with a tolerable drop in F-1 score.

BLAZEIT. On Q1, BLAZEIT outperforms other systems with respect to query processing time. However, as we discussed in §3, its specialized model delivers a lower F-1 score. On other queries, since the F-1 score of the specialized model is too low to be useful, BLAZEIT falls back to the oracle model. Even though the specialized model is not effective, BLAZEIT still evaluates the query with the specialized model, so the processing time of BLAZEIT is higher than that of NAIVE for Q2, Q3, and Q4.

PP. PP reduces the processing time on Q3 and Q4. This is because these two queries focus on relatively rare events. So, the model in PP is able to filter out a significant fraction of frames to accelerate query processing.

7.3 Ablation Study

We next examine how the EPs in a model are used by THIA-EI while processing queries. The results of this experiment are shown

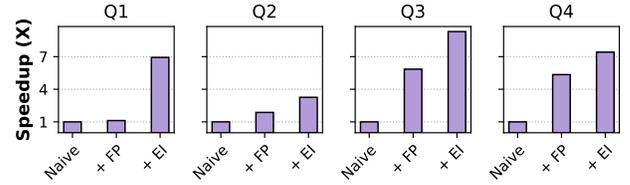


Figure 11: Ablation study – Contribution of FINE-GRAINED PLANNING and EARLY INFERENCE techniques to the performance of THIA-EI.

in Figure 10. To better understand the contribution of each technique to the performance of THIA-EI, we also conduct an ablation study. We measure the execution time of THIA-SINGLE and THIA-EI to illustrate the benefits of FINE-GRAINED PLANNING and EARLY INFERENCE techniques, respectively. The results of this study are illustrated in Figure 11.

These two experiments demonstrate that: (1) FINE-GRAINED PLANNING is able to adaptively choose the appropriate EP for each chunk based on the query and video frames, and (2) FINE-GRAINED PLANNING and EARLY INFERENCE techniques have significant impact for rare events; but, in the case of frequent events, the speedup mainly comes from EARLY INFERENCE.

On Q1, since positive events appear in majority of the video frames, the impact of FINE-GRAINED PLANNING is minimal. When we add in the EARLY INFERENCE technique, THIA-EI delivers higher speedup by using shallow EPs for easy-to-detect events, as shown in Figure 10. Q1 demonstrates an extreme scenario wherein the first EP (EP-1) provides correct predictions on all video frames. In contrast, in the case of Q2, THIA-EI must use multiple EPs due to harder-to-detect events. Here, the OPTIMIZER reduces the execution time by carefully choosing the EPs to use. As shown in Figure 10, while some frames are assigned to the oracle EP, other frames are assigned to shallow EPs to reduce execution time. By reducing the execution time, THIA-EI delivers a 4× speedup over NAIVE.

Unlike queries focusing on frequent events, the FINE-GRAINED PLANNING technique provides more performance benefits in the case of queries related to rare events, because the OPTIMIZER decides to skip some chunks during execution (Line 11). On Q3 and Q4, as shown in Figure 11, FINE-GRAINED PLANNING leads to a 5× speedup. Nevertheless, the EARLY INFERENCE technique is still useful for these queries. THIA-EI carefully assigns certain video frames to shallow EPs to improve the performance without losing accuracy. The system is thus accelerated further by 3× when the EARLY INFERENCE technique enabled.

7.4 Impact of FINE-GRAINED PLANNING

We demonstrate the benefits of FINE-GRAINED PLANNING by comparing THIA-EI against a system that uses coarse-grained planning with the same EARLY INFERENCE model. With coarse-grained planning, we evaluate all EPs on 10% of the sampled video frames and pick the EP that meets the precision and recall constraints. We show a breakdown of the query processing time in Figure 12. On all queries, FINE-GRAINED PLANNING provides a better query plans than coarse-grained planning so that the execution time is consistently lower. Moreover, with optimizations like sampling rate bounds and memoization in FINE-GRAINED PLANNING, it does

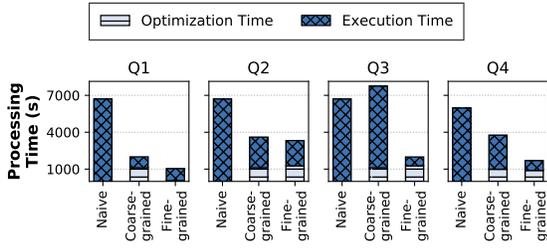


Figure 12: Impact of FINE-GRAINED PLANNING – Breakdown of query processing time with fine-grained and coarse-grained planning techniques.

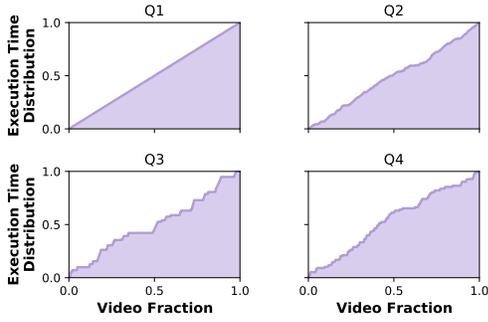


Figure 13: Variation of Execution Time – Variation of query execution time across the chunks in the video.

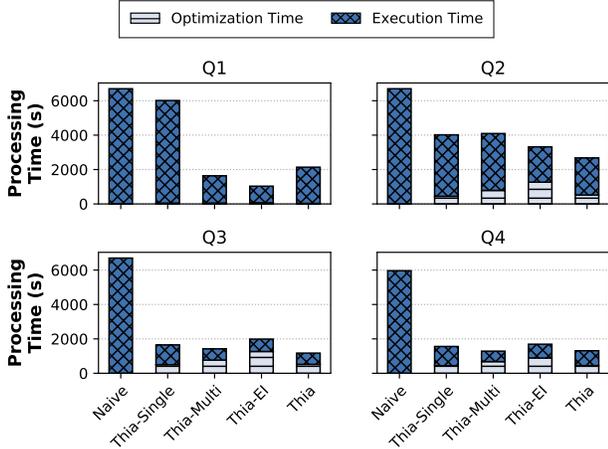


Figure 14: Breakdown of query processing time – Components of query processing time (optimization time and execution time) associated with NAIVE, THIA-SINGLE, THIA-MULTI, THIA-EI, and THIA.

not incur higher optimization time than the naive coarse-grained planning approach.

We next measure the distribution of query execution time over the fraction of the video being analysed in Figure 13. An even distribution (e.g., Q1 and Q2) suggests that the same query plan is used for a large chunk. In contrast, an uneven distribution (e.g., Q3 and Q4) suggests that the plan changes frequently across the video.

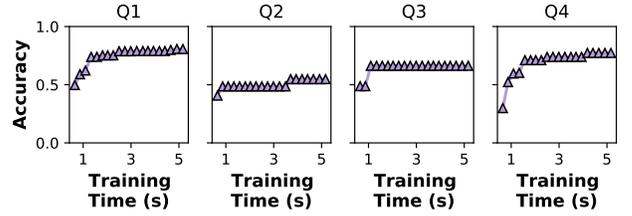


Figure 15: Accuracy of the EXIT POINT ESTIMATION model – Variation in validation accuracy over training time.

Query	Under (%)	Over (%)
Q1	0.00	27.29
Q2	20.29	21.42
Q3	28.96	9.64
Q4	7.20	14.40

Table 5: Impact of EXIT POINT ESTIMATION technique – Accuracy of of EXIT POINT ESTIMATION technique relative to THIA-EI.

7.5 Processing Time Breakdown

We now provide a breakdown of the processing time of THIA-EI and its variants and compare it against NAIVE. Recall that all systems except for NAIVE use the FINE-GRAINED PLANNING technique. Though THIA-SINGLE is only able to use the oracle EP, it is able to skip frames with no relevant events using FINE-GRAINED PLANNING.

The results are shown in Figure 14. Access to a set of EPs allows both THIA-EI and THIA-MULTI to reduce execution time in comparison to THIA-SINGLE and NAIVE. The reduction in execution time is more prominent compared to THIA-SINGLE for queries focusing on more frequent events. While THIA-MULTI supports multiple EPs similar to THIA-EI, THIA-EI supports multiple EPs in a single model. So, it has a lower GPU memory footprint, as shown in Figure 18. In addition to that, THIA-EI offers more flexibility in terms of creating and selecting different EPs. On Q2, due to the limited flexibility of THIA-MULTI, it has lower execution time and also lower accuracy than THIA-EI.

The cons of using multiple EPs with FINE-GRAINED PLANNING is the increase in optimization time. This is because the OPTIMIZER has to evaluate all EPs to choose an optimal EP. As illustrated in Figure 14, the optimization time of THIA-EI and THIA-MULTI is consistently higher than that of THIA-SINGLE. Increasing this flexibility (i.e., adding more EPs) leads to higher sampling overhead. Thus, THIA-EI has higher optimization time than THIA-MULTI and both have higher optimization time than THIA-SINGLE. This motivates the need for reducing the optimization time.

7.6 Impact of EXIT POINT ESTIMATION

TRAINING TIME. Since the OPTIMIZER needs to train an estimation model for every unique query, we first quantify the training overhead of the EXIT POINT ESTIMATION technique. Figure 15 shows the variation in validation accuracy over training time. The model quickly converges since the THIA uses a small training set (200 samples) and a two-layer neural network for EXIT POINT ESTIMATION.

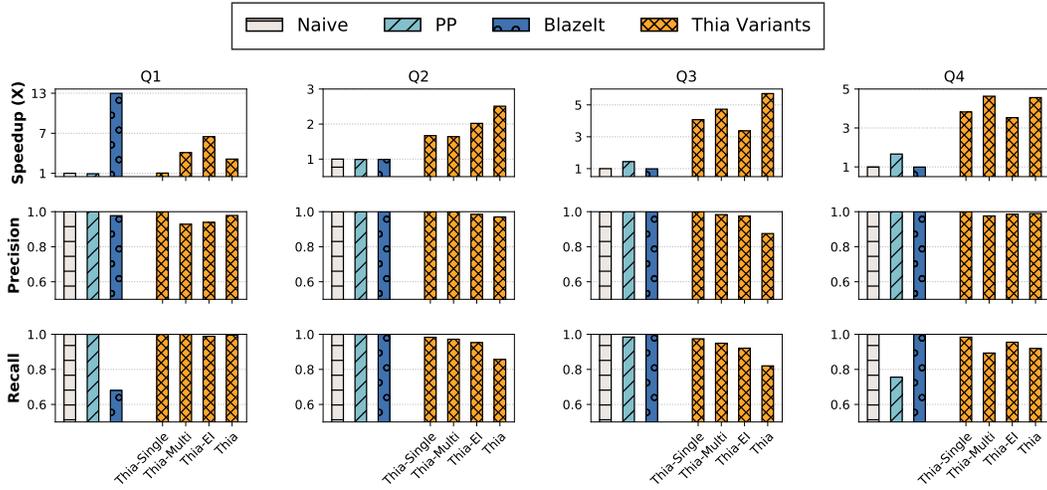


Figure 16: End-to-end Comparison – Comparative analysis of speedup, precision, and recall metrics against state-of-the-art video analytics systems.

It takes less than 5 seconds (0.1% of total processing time) to train each of these models for all queries.

OPTIMIZATION TIME. We next investigate the efficacy of the EXIT POINT ESTIMATION technique in reducing the optimization time. We integrate the EXIT POINT ESTIMATION technique into THIA-EI to construct THIA. Using the EXIT POINT ESTIMATION technique, THIA is able to directly predict an appropriate EP to use for a chunk. In contrast, THIA-EI runs inference using all EPs during optimization phase to select the EP. As shown in Figure 14, THIA cuts the optimization time in half. Recall that THIA uses the object detection EPs for FINE-GRAINED PLANNING. So, the EXIT POINT ESTIMATION technique uses the backbone features for choosing a plan for each chunk. We also measure the overhead of using EXIT POINT ESTIMATION. This technique introduces a minimal additional overhead (18 s) even under the highest sampling rate (processing time is in order of thousands of seconds).

EXECUTION TIME. Lastly, we discuss the impact of the EXIT POINT ESTIMATION technique on planning accuracy (*i.e.*, choosing the optimal EP) and execution time. We measure the planning accuracy relative to THIA-EI. In Table 5, Under and Over represent the percentage of chunks for which the EXIT POINT ESTIMATION technique returns a shallower EP and a deeper EP than that returned by THIA-EI. While shallower estimates hurt query accuracy, deeper estimates increase execution time. As shown in Figure 14, execution time increases only negligibly for all queries except for Q1. Since the EXIT POINT ESTIMATION technique reduces optimization time, the total processing time of THIA is lower than that of THIA-EI on all queries except for Q1. This is because Q1 can be accurately answered using the first EP (Figure 10), so deeper estimates increase execution time. We discuss the impact on query accuracy in §7.7.

7.7 End-to-End Comparison

We report the speedup, precision, and recall metrics with respect to other state-of-the-art systems in Figure 16. The bars on the left side represent three systems: (1) NAIVE, (2) BLAZEIt, and (3) PP. The

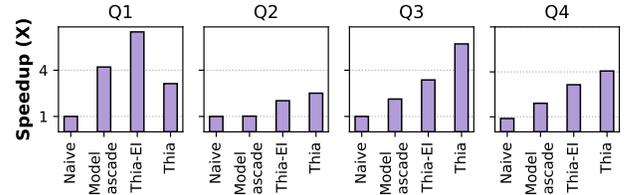


Figure 17: Model Cascade vs. EARLY INFERENCE – Comparison of the query processing time taken by NAIVE, MODEL CASCADE, THIA-EI, and THIA.

latter two video analytics systems use specialized models. The bars on the right side represent variants of THIA that use one or more techniques presented in this paper.

The most notable observation is that systems that have access to a set of EPs deliver higher performance than those that have access to a single EP. Unlike THIA-MULTI, which maintains a collection of separate models, the EARLY INFERENCE technique offers more flexibility in choosing the optimal EP. However, this technique increases the optimization time. We overcome this limitation using the EXIT POINT ESTIMATION technique.

THIA consistently delivers higher speedup than other systems. Due to the inaccuracy of the EXIT POINT ESTIMATION technique, THIA has a minimal drop in accuracy. The drop in recall is more prominent because shallow EPs are unable to recognize hard-to-detect events, which leads to more false negatives. In contrast, the drop in precision is minimal. On Q1, THIA improves both precision and recall. This is because the OPTIMIZER augmented with the EXIT POINT ESTIMATION technique overestimates the EPs for this query, leading to an improvement in accuracy.

7.8 Model Cascade vs. EARLY INFERENCE

As we mentioned in §2.3, THIA-MULTI uses a model cascade [1]. It differs from the naive model cascade technique in that it uses FINE-GRAINED PLANNING to select EPs. In contrast, a naive approach

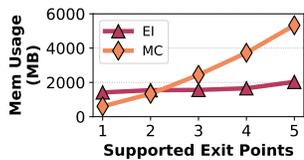


Figure 18: Memory Footprint – Comparison of memory footprint of THIA-EI and THIA-MULTI (*i.e.*, a model cascade).

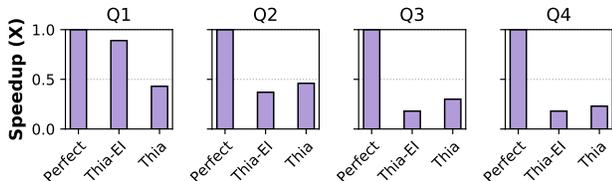


Figure 19: Optimality of FINE-GRAINED PLANNING – Comparison of execution time of THIA-EI and THIA against that with the optimal plan.

determines whether to stop at an EP based on the a confidence score of the prediction from the previous EP. THIA-MULTI outperforms the naive approach since shallower EPs in the model cascade are always executed with the latter technique. As a result, our EARLY INFERENCE based systems (*i.e.*, THIA-EI and THIA) outperform the model cascade approach. Since it is challenging to construct a confidence-score based system in the case of object detection, we show the projected performance of the model cascade approach compared to THIA-EI and THIA in Figure 17. The EARLY INFERENCE based system delivers 2× speedup compared to MODEL CASCADE.

Using multiple models to construct model cascade also increases the memory footprint of the system. As shown in Figure 18, the real-time memory usage of a model cascade increases when we increase the number of EPs. It has a 5 GB memory footprint with 5 EPs. In contrast, since the EARLY INFERENCE model shares parameters and inference features, it only incurs a 2 GB memory footprint for the same number of EPs.

7.9 Optimality of FINE-GRAINED PLANNING

We now examine the quality of the query plans relative to the optimal plan by comparing the execution speedup. The optimal plan is constructed using a brute-force EP selection on every frame (*i.e.*, chunk size = 1). The optimal plan is un-achievable in reality because the brute-force selection on every frame significantly increases optimization time. The results are shown in Figure 19. The plans constructed by the OPTIMIZER are 0.3× slower than the optimal plan. So, there is still potential for improving the quality of the query plans. We plan to explore techniques for doing so with a tolerable impact on optimization time in the future.

8 Limitations

ACCURACY OF EXIT POINT ESTIMATION MODEL. The EXIT POINT ESTIMATION technique uses a simple neural network to model the optimal EPs distribution. However, the inaccuracy of this model

leads to a minimal drop in overall query accuracy. We plan to study other techniques to reduce the optimization time in the future. For example, instead of using a deep learning model, a lightweight statistical estimator may be sufficient. A challenge with this approach is that this estimator must accurately map all of the parameters returned by the object detection model (*e.g.*, a set of bounding boxes and confidence scores) to the appropriate EP.

QUERY SUPPORT. Currently, THIA supports a limited set of queries. To support general-purpose video analytics, we will need to add support for additional types of queries (*e.g.*, aggregate queries). We plan to integrate the EARLY INFERENCE and FINE-GRAINED PLANNING techniques into the query execution engine and the query optimizer of a full-featured video analytics system in the future.

9 Related Work

MODEL CASCADE. Researchers in the area of face detection have proposed models that support a set of EPs that are geared for different accuracy and speed trade-offs [19, 20]. These models return a binary decision and a confidence score (*i.e.*, whether a face exists). Based on the confidence score, the model chooses the appropriate EP. In contrast, THIA uses the estimator to directly pick the EP.

QUERY PLANNING. The authors of Chameleon [9] observe that an appropriate query plan is critical to gain high performance and accuracy. Similar to the FINE-GRAINED PLANNING technique, it adjusts the execution plan at runtime. To reduce the cost of picking the correct plan, it exploits temporal locality of nearby frames in the video, thereby reducing the profiling cost. To further reduce this cost, it uses a clustering algorithm to explore correlation across videos. THIA instead uses a shallow neural network to directly estimate the optimal EP to use for a chunk.

10 Conclusion

We presented, THIA, a video analytics system for efficiently processing visual data at scale. THIA leverages the early inference technique to support a range of throughput-accuracy tradeoffs. It then adopts a fine-grained approach to query planning and processes different chunks of the video with different exit points to meet the user’s requirements. Lastly, THIA uses a lightweight technique for directly estimating the exit point using a shallow deep learning model to lower the optimization time. We empirically show that these techniques enable THIA to outperform two state-of-the-art video analytics systems by up to 6.5×, while providing accurate results even on queries focusing on hard-to-detect events.

References

- [1] Michael R. Anderson, Michael Cafarella, German Ros, and Thomas F. Wenisch. 2019. Physical Representation-Based Predicate Optimization for a Visual Analytics Database. *ICDE* (2019).
- [2] Favyen Bastani, Songtao He, Arjun Balasingam, Karthik Gopalakrishnan, Mohammad Alizadeh, Hari Balakrishnan, Michael Cafarella, Tim Kraska, and Sam Madden. 2020. MIRIS: Fast Object Track Queries in Video. *SIGMOD* (2020).
- [3] P Baumann, A Dehmel, P Furtado, R Ritsch, and N Widmann. 1998. The Multidimensional Database System RasDaMan. *SIGMOD* (1998).
- [4] Christopher Canel, Thomas Kim, Giulio Zhou, Conglong Li, Hyeontaek Lim, David G Andersen, Michael Kaminsky, and Subramanya R Dullloor. 2019. Scaling Video Analytics on Constrained Edge Nodes. *SysML* (2019).
- [5] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2018. Mask R-CNN. *ICCV* (2018).
- [6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. *CVPR* (2015).
- [7] Kevin Hsieh, Ganesh Ananthanarayanan, Peter Bodik, Shivaram Venkataraman, Paramvir Bahl, Matthai Philipose, Phillip B Gibbons, and Onur Mutlu. 2018. Focus: Querying Large Video Datasets with Low Latency and Low Cost. *OSDI* (2018).
- [8] Ramesh Jain and Arun Hampapur. 1994. Metadata in Video Databases. *SIGMOD* (1994).
- [9] Junchen Jiang, Ganesh Ananthanarayanan, Peter Bodik, Siddhartha Sen, and Ion Stoica. 2018. Chameleon: scalable adaptation of video analytics. *SIGCOMM* (2018).
- [10] Daniel Kang, Peter Bailis, and Matei Zaharia. 2019. BlazeIt: Optimizing Declarative Aggregation and Limit Queries for Neural Network-Based Video Analytics. *VLDB* (2019).
- [11] Daniel Kang, John Emmons, Firas Abuzaid, Peter Bailis, and Matei Zaharia. 2017. NoScope: Optimizing Deep CNN-Based Queries over Video Streams at Scale. *VLDB* (2017).
- [12] Yao Lu, Aakanksha Chowdhery, and Srikanth Kandula. 2016. Optasia: A Relational Platform for Efficient Large-Scale Video Analytics. *SoCC* (2016).
- [13] Yao Lu, Aakanksha Chowdhery, Srikanth Kandula, and Surajit Chaudhuri. 2018. Accelerating Machine Learning Inference with Probabilistic Predicates. *SIGMOD* (2018).
- [14] M-E. Nilsback and A. Zisserman. 2008. Automated Flower Classification over a Large Number of Classes. *ICVGIP* (2008).
- [15] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. *NIPS-W* (2017).
- [16] Luis Remis, Vishakha Gupta-Cledat, Christina Strong, and Ragaad Altarawneh. 2018. VDMS: Efficient Big-Visual-Data Access for Machine Learning Workloads. (2018).
- [17] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. 2015. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *NeurIPS* (2015).
- [18] Karen Simonyan and Andrew Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. *ICLR* (2015).
- [19] Yi Sun, Xiaogang Wang, and Xiaoou Tang. 2013. Deep Convolutional Network Cascade for Facial Point Detection. *CVPR* (2013).
- [20] P. Viola and M. Jones. 2001. Rapid Object Detection Using a Boosted Cascade of Simple Features. *CVPR* (2001).
- [21] Longyin Wen, Dawei Du, Zhaowei Cai, Zhen Lei, Ming-Ching Chang, Honggang Qi, Jongwoo Lim, Ming-Hsuan Yang, and Siwei Lyu. 2020. UA-DETRAC: A New Benchmark and Protocol for Multi-Object Detection and Tracking. *CVIU* (2020).
- [22] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. 2019. Detectron2. <https://github.com/facebookresearch/detectron2>.
- [23] Haoyu Zhang, Ganesh Ananthanarayanan, Peter Bodik, Matthai Philipose, Paramvir Bahl, and Michael J Freedman. 2017. Live Video Analytics at Scale with Approximation and Delay-Tolerance. *NSDI* (2017).
- [24] Yuhao Zhang and Arun Kumar. 2019. Panorama: A Data System for Unbounded Vocabulary Querying over Video. *VLDB* (2019).