Marshalling Model Inference in Video Streams

Daren Chao
University of Toronto
drchao@cs.toronto.edu

Nick Koudas
University of Toronto
koudas@cs.toronto.edu

Xiaohui Yu
York University
xhyu@yorku.ca

***Abstract*—Numerous cloud platforms are available to deploy and train deep models as well as process data, such as Amazon Rekognition and Azure custom Vision Service, which have made it easy for companies to adopt deep learning technologies in their operations. Commonly such services price usage per image or frame in typical applications that consume video streams and as a result the costs rapidly accumulate. In this paper we introduce a model, named EventHit, that is able to marshal model inference requests in such services by making predictions over the video stream about events of interest. As such only relevant video segments are sent for analysis to the cloud infrastructure and irrelevant parts are filtered from further processing. We introduce the architecture and fully describe its components. We present two novel optimizations in this context that aim to provide control over the trade-off between prediction accuracy (especially regarding the probability of missing an event of interest) and processing cost at the cloud infrastructure. We fully describe and analyze our proposals in the context of real datasets. We also present the results of a detailed experimental evaluation varying parameters of interest and demonstrate the practical utility of our proposals.**

I. Introduction

The rapid advances in Deep Learning (DL) [1] have revolutionized numerous applications of vast practical significance such as various aspects of video analytics (including video object detection [2], event detection [3], action recognition [3] and classification [4] to name a few). The prevalence of DL frameworks (such as TensorFlow and Pytorch) has rendered access to powerful models for advanced deep learning a commodity. Access to powerful cloud-enabled commodity cameras [5] is now within reach for any business that wishes to accumulate and analyze video data (in real-time or store them for future reference effortlessly in the cloud). Applications such as video surveillance automation, sports analytics, news clip analysis and autonomous driving are thriving.

Although access to video data and the associated models to analyze them are both within reach of practitioners, assembling the technical expertise to build and maintain the required infrastructure for analysis as well as the technical know-how to fine tune and optimize models is still highly challenging. Although businesses are ready to make use of the results of analysis (e.g., adopt intelligent video surveillance in their operations), it is challenging to build all this expertise in house. For this reason most businesses will opt to outsource the complexities of infrastructure and associated technology and focus on the unique aspects of their application. Numerous such services are offered by leading cloud providers such as Amazon Rekognition [6], Azure custom vision service [7], Google Cloud Vision API [8], IBM Watson Visual Recognition [9], Alibaba Cloud intelligence Vision [10] as well as more specialized players such as Clarifai [11] among many others. The main functionality of such services is to offer the infrastructure and the associated deep models as a service. That way, users focus on their specific applications without bothering with ownership and maintenance of the infrastructure and associated model development/maintenance and training. Instead users connect their data sources, i.e., video feeds in the case of video analytics, specifying the desired functionality (e.g., detection of specific events in the case of video surveillance) and the suitable models produce the desired output.

Commonly, such services price usage on a per frame basis, and in typical applications that consume video streams, the costs rapidly accumulate. It is evident that depending on the application, there is occasionally a "needle in a haystack" scenario in which a great deal of service is over-utilized (with associated costs) as only a fraction of frame sequences would be of interest; the rest of the video stream is irrelevant. For example, in a surveillance automation application that focuses on specific events (e.g., trucks approaching a gate at a construction site, triggering an automated gate opening), only a fraction of frames would be periodically relevant (those that contain the event). Depending on the application, events may be identically and independently distributed (i.i.d.), such as Poisson as in the case of truck arrivals [12]. The cloud service has to consume frames constantly to detect the event of interest however, resulting in excessive costs. Examples like this abound in video surveillance automation, especially in industrial automation (e.g., recognizing defective products in industrial pipelines, which may be i.i.d. based on a Poisson or geometric distribution [13], and triggering automated removal [14]) and large commercial spaces (e.g., recognizing crowding due to rapid arrivals and throttling them [14]).

In this paper, we propose a framework EventHit to learn to predict when events of interest, following an underlying but unknown distribution, occur. Such prediction guides our decision to direct for analysis on the portion of the video stream likely to contain events of interest (incurring the associated monetary costs). For parts of the stream that no events are predicted to occur we may skip the associated processing.

Figure 1 presents an overview of the proposed architecture. An array of smart cameras records video, relaying the feeds to EventHit. A prediction is conducted for a temporal event horizon, namely a temporal window of frames, up to some adjustable maximum time into the future, given the current

point in time (current set of frames). The prediction consists of a range of frames into the future that is most likely to contain an event of interest. The suitable range of frames is relayed to the cloud service and the associated infrastructure of choice (CI) for further processing (to conclusively, based on deep models, assess if the event is present) and the final results are routed to the application. EventHit can reside on premise or in the cloud. Training EventHit takes place before its deployment and is conducted once by routing the video stream to CI and collecting the output of the cloud-based service models for the events of interest. This constitutes the training data for any set of events of interest. After training, the video streams are routed to EventHit first; a prediction is conducted for the range of frames of the next event occurrence. The range of frames in the prediction are relayed to the CI for subsequent processing (as detailed in § III).

To instantiate EventHit we propose a deep neural model that utilizes features from the video frames as well features extracted from the temporal dynamics of the video along with a family of event-specific sub-networks to handle predictions for each event of interest. Predictions occur in an adjustable temporal event horizon. We fully describe the architecture of our network which is trained end-to-end for the specific events.

We further present two novel optimizations in the context of this problem, namely C-CLASSIFY and C-REGRESS, to provide control over the trade-off between prediction accuracy (especially regarding the probability of missing an event of interest) and processing cost by the CI. Both optimizations are grounded on the theory of conformal predictions [15], [16]. The basic idea of C-CLASSIFY is to calibrate the predictions of EventHit using a set of predictions and associated ground truth as a reference. Based on this reference set, it builds a probabilistic framework around how *similar* a new prediction is with respect to the reference set. That way C-CLASSIFY builds a quantifiable and adjustable trade-off between precision and recall in predicting the existence of an event in the temporal event horizon. The desired recall can be adjusted in a probabilistic manner to a desired level (effectively controlling accuracy of the predictions) incurring an associated adjustment in the precision.

If an event of interest is predicted to take place in the next temporal horizon by C-CLASSIFY, EventHit will predict an interval of frames that such an event is predicted to occur. To tune the accuracy of such a prediction in a quantifiable manner, C-REGRESS builds temporal ranges around the start and end frames in this predicted interval respectively, with specific probabilistic accuracy guarantees. Such temporal ranges essentially correspond to a range of frames around the predicted start and end frames. Taking these frame ranges into account, we adjust the prediction of EventHit effectively establishing probabilistic trade-offs between recall and excess frames processed.

In this paper we make the following contributions: 1) we introduce the novel problem of marshalling model inferences in video streams; 2) we propose a novel deep architecture, called EventHit, inspired by survival analysis [17], [18] to conduct

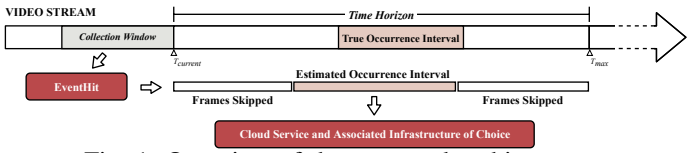


Fig. 1: Overview of the proposed architecture.

predictions for multiple events of interest; 3) we establish trade-offs between the accuracy of event detection and the cost of detection when cloud ML infrastructures are utilized; 4) we present a thorough experimental evaluation utilizing real videos, that validate the results of our analysis and demonstrate the efficacy of our proposals in a practical setting.

II. Problem Definition

A video stream is a sequence of frames $V = \langle f_1, \ldots, f_N \rangle$, where f_i is a frame at timestamp T_i, and N is the length of the video stream. The number of frames (length) of a video N can be fixed or unbounded. Let $\mathcal{E} = \{E_1, \ldots, E_k\}$ be a set of k independent event types of interest in the video stream where an event is a complex activity that is localized in time and space. An event is expected to be observable in the video, involving interactions among people and/or objects [19]–[21]. Such event types are typically application-dependent. In our video surveillance automation example, an event type of interest can be a truck approaching the gate in an industrial setting. Note that multiple instances of an event type can take place in the stream (i.e., multiple events corresponding to trucks approaching a gate taking place at different points in time).

One basic assumption is that for each of the event types $E_i \in \mathcal{E}$, a CI of choice provides access to a model of high accuracy to detect them. An *occurrence interval* (OI) of an event instance in a stream is a time interval $(T^s, \ldots, T^e)$ in which the event instance occurs, with T^s and T^e being the start and end timestamps respectively.

Our goal is to build a lightweight and general model (i.e., a model with structure applicable to different event prediction challenges) such that at any time T_i we can predict the occurrence interval of an event instance belonging to event type $E_i \in \mathcal{E}$ using information collected on the frames up to f_i, referred to as *covariates*. We constrain our event occurrence predictions up to and including an (adjustable) finite number of frames in the future that we refer to as the *time horizon* (H). To ease notation, we assume that event instances of $E_i \in \mathcal{E}$ can appear at most once in the time horizon for estimation purposes. We emphasize however that our entire framework and solution can address the case of multiple event instances of the same event type taking place in the time horizon[1]. For this reason, in the sequel we will refer to an event type $E_i \in \mathcal{E}$ and its instances as *events* interchangeably.

Covariates are part of feature selection and are application-dependent. We will extract features from each frame and combine them to form a feature vector. The features can be extracted from a single frame (e.g., presence or absence of

[1]This involves modifying the structure of EventHit outlined in § III so that each sub-network corresponding to an event type makes multiple predictions as opposed to one.

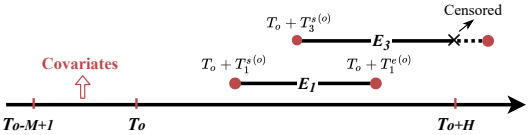


Fig. 2: An illustration demonstrating the task of predicting the next occurrence interval of events.

certain objects in a frame) or multiple frames (e.g., relative object positions in the frame sequence, estimates of object speed and relative object distance). Without loss of generality, we assume that there are D feature types and thus the dimensionality of each feature vector is D. To construct the covariates $\mathcal{X}_i$ at time T_i, we utilize feature vectors collected from a sequence of *consecutive* frames, a *collection window* W of length $M = |W|$. Then the covariates $\mathcal{X}_i \in R^{M\times D}$ can be written as:

$$\mathcal{X}_i = [X_{i-M+1}, \ldots, X_i],$$

where X_m is a feature vector extracted from frame f_m. At any frame f_i, we extract the covariates $\mathcal{X}_i$ and observe the events that will occur in the next time horizon, H, along with their associated occurrence intervals. Let $\mathcal{L}_i$ be a set of events that occur in the time horizon starting from frame f_i, $\mathcal{L}_i \subseteq \mathcal{E}$. More than one event may occur within a time horizon H. Let $\mathcal{T}_i$ represent the occurrence intervals of the events in set $\mathcal{L}_i$,

$$\mathcal{T}_i = \left\{ \left[T_k^{s(i)}, T_k^{e(i)} \right] : \forall\ E_k \in \mathcal{L}_i \right\},$$

where $T_k^{s(i)}, T_k^{e(i)} \in [1, H]$ represent timestamp offsets of the start and end frames of the occurrence interval of event E_k relative to T_i. Thus, the absolute start and end timestamps of frames for event k at $\mathcal{T}_i$ are $T_k^{s(i)} = T_i + T_k^{s(i)}$ and $T_k^{e(i)} = T_i + T_k^{e(i)}$, respectively.

The triplets $(\mathcal{X}_i, \mathcal{L}_i, \mathcal{T}_i)$ can be generated for any frame f_i (except the first M frames in V) in the video stream. Figure 2 presents an example. For a given timestamp T_o, covariates are extracted from a collection window of size M with frames corresponding to timestamps from T_{o-M+1} to T_o. We observe that some events (such as Event E_1) start within the time horizon and end before $T_o{+}H$; while others (Event E_3) end after the end of the time horizon. We call such events *censored* events. Let δ_k be an indicator variable representing whether an event E_k is censored ($\delta_k = 1$ if E_k is censored; $\delta_k = 0$ otherwise), and Γ_i be the set of corresponding indicators for events in $\mathcal{L}_i$. For the example shown in Figure 2,

$$\mathcal{L}_o = \{E_1, E_3\},$$
$$\Gamma_o = \{\delta_1, \delta_3\} = \{0, 1\},$$
$$\mathcal{T}_o = \{[T_1^{s(o)}, T_1^{e(o)}], [T_3^{s(o)}, T_3^{e(o)}]\},$$

where $T_k^{s(o)} \in [1, H]$ and $T_k^{e(o)} \in [1, H]$ represent the start and end time of the occurrence interval of event E_k. When considering events within the time horizon of a frame f_i at T_i, for an event E_k that is censored, we set the end of the occurrence interval for this event as the end time of the time horizon for T_i. In our example for T_o, since $\delta_3 = 1$, we have $T_k^{e(o)} = H$.

We build a training dataset by sampling frames from the beginning of the video stream (frames f_1 to f_P) and extracting triplets (namely records). Let $\mathcal{D}_{\text{train}}$ represent the set of records for training and $\mathcal{P}_{\text{train}}$ be the set of frames selected from frames f_1 to f_P,

$$\mathcal{D}_{\text{train}} : \{(\mathcal{X}_n, \mathcal{L}_n, \mathcal{T}_n)\}_{n\in\mathcal{P}_{\text{train}}}. \quad (1)$$

Given the training data collected from a video stream, the problem of interest in this paper is to build a prediction model to predict, at any $T_j > T_P$, whether and when the events in $\mathcal{E}$ will occur in the time horizon, T_j to $T_j{+}H$. In comparison to the services offered by the CI, the model built to predict whether and when events will occur is intended to be lightweight, general and easy to deploy locally.

III. EventHit

We propose a framework called EventHit to make predictions on upcoming occurrences of events in $\mathcal{E}$. To realize things concrete, given a collection of $|\mathcal{P}|$ training records, we develop a particular instantiation of the framework that utilizes a Neural Network model to yield the predictions. Our goal would be to use the network as a proxy to estimate the following unknown probabilities for each event E_k:

- The probability that an event E_k will occur within the next time horizon given observed covariates $\mathcal{X}_n$,
$$P\left(E_k \in \mathcal{L}_n \mid \mathcal{X} = \mathcal{X}_n\right). \quad (2)$$
- The probability that event E_k occurs at $T_j \in [0, H]$,
$$P\left(T_j \in [T_k^{s(n)}, T_k^{e(n)}] \mid \mathcal{X} = \mathcal{X}_n, E_k \in \mathcal{L}_n\right), \quad (3)$$
where $[T_k^{s(n)}, T_k^{e(n)}] \in \mathcal{T}_n$.

To the best of our knowledge, this is the first time a neural network model has been proposed to achieve both Boolean predictions of *if* the event will occur and range predictions of *when* the event will occur.

The proposed DNN model is depicted in Figure 3. EventHit consists of a shared sub-network and K event-specific sub-networks, one for each event in $\mathcal{E}$. The shared sub-network accepts covariates $\mathcal{X}_n \in R^{M\times D}$ as input and produces a vector z that captures the latent representation as output. It first utilizes a Long Short Term Memory (LSTM) [22] encoder that is suitable for modeling temporal relationships in the video stream across frames. The LSTM encoder processes the feature vectors $(X_m{\in}\mathcal{X}_n, n{-}M{<}m{\leq}n)$ in sequence, updating corresponding hidden states at each time-step: $h_m{=}\text{LSTM}(h_{m-1}, X_m)$. The last hidden state output from the LSTM, h_n, is directed to the fully connected and dropout layer(s). Then the hidden vector z is concatenated with X_n, i.e., the last feature vector in covariates $\mathcal{X}_n$.

Each event-specific sub-network consists of the fully connected layer(s) with independent weights and is activated by a sigmoid function[2]. Each sub-network accepts the concatenated hidden state $z \oplus X_n$ as input, and produces an output vector consisting of coordinates with values as proxies for the probabilities in Equations (2) and (3). Specifically, for event E_k, the corresponding sub-network outputs a vector $\Theta_k^{(n)}{=}\left[b_k^{(n)}, \theta_{k,1}^{(n)}, \theta_{k,2}^{(n)}, ..., \theta_{k,H}^{(n)}\right]$, where $b_k^{(n)}{\in}[0, 1]$ represents a score to quantify the occurrence of event E_k in the time horizon, and $\theta_{k,v}^{(n)} \in [0, 1]$ represents a score to quantify the

[2]We choose the sigmoid layer since the event existence prediction for each event can be seen as a binary classification problem.

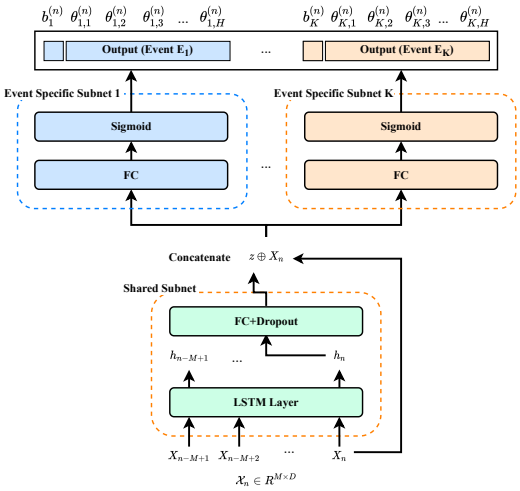


Fig. 3: The architecture of EventHit.

occurrence of event E_k in the frame with offset v from T_n, i.e., f_{v+n}.

The network is trained end-to-end utilizing the vectors $\Theta_k^{(n)}$. In particular, the loss functions are a) the average cross-entropy loss between the estimated binary class score and the ground truth class (where the two classes for an event E_k correspond to whether E_k happens or not)

$$L_1 = \frac{-1}{|\mathcal{P}|} \sum_{n \in \mathcal{P}} \sum_{k=1}^{K} \beta_k \left(\mathbb{1}[E_k \in \mathcal{L}_n] \log b_k^{(n)} + \mathbb{1}[E_k \notin \mathcal{L}_n] \log(1 - b_k^{(n)}) \right);$$

where $\mathbb{1}$ is an indicator function and β_k represents the weight of classification loss of the event E_k; and b) the average cross-entropy loss between the estimated score for the occurrence of event E_k and the ground truth label for each frame in the time horizon representing whether E_k does occur in that frame,

$$L_2 = \frac{-1}{|\mathcal{P}|} \sum_{n \in \mathcal{P}} \sum_{k=1}^{K} \gamma_k \mathbb{1}[E_k \in \mathcal{L}_n] \left[\sum_{T_v \in [T_k^{s(n)}, T_k^{e(n)}]} \frac{\log \theta_{k,v}^{(n)}}{T_k^{e(n)} - T_k^{s(n)}} + \sum_{T_v \in [1,H] \setminus [T_k^{s(n)}, T_k^{e(n)}]} \frac{\log(1 - \theta_{k,v}^{(n)})}{H - (T_k^{e(n)} - T_k^{s(n)})} \right],$$

where γ_k represents the weight of occurrence prediction loss of the event E_k. The hyper-parameters β_k and γ_k, for all $E_k \in \mathcal{E}$, can be tuned by grid search [23], [24]. To train EventHit, we minimize a total loss function L_{Total} that is the sum of the two losses, $L_{\text{Total}} = L_1 + L_2$.

For training, we utilize $\mathcal{D}$: $\{(\mathcal{X}_n, \mathcal{L}_n, \mathcal{T}_n)\}_{n \in \mathcal{P}}$ in order to compute the loss corresponding to input covariates and train the network end-to-end. The choice of covariates $\mathcal{X}_n \in R^{M \times D}$ is application-dependent. Like any other application of ML, this is a task that requires feature engineering. We select features through standard correlation analysis methods [25]. Other feature engineering approaches can be utilized in this stage, such as dimensionality reduction [26] via auto-encoders [27]. In § VI we will detail the covariates utilized for our target applications.

At inference time, the vector $\Theta_k^{(n)}$ produced by the event-specific sub-network is utilized for predictions. We choose a threshold τ_1 (e.g., 0.5) to predict the existence of event $E_k \in \mathcal{E}$ in the time horizon. With a $b_k^{(n)}$ no less than τ_1, event E_k is predicted to occur in the time horizon from T_n,

$$b_k^{(n)} \geq \tau_1 \Rightarrow \mathbb{1}[E_k \in \hat{\mathcal{L}}_n] = 1, \tag{4}$$

where $\hat{\mathcal{L}}_n$ represents the estimated set[3] of events that will occur in the time horizon from T_n. Upon determining that event E_k is predicted to occur, we derive an estimate of its occurrence interval $\left[\hat{T}_k^{s(n)}, \hat{T}_k^{e(n)}\right]$ in the time horizon. We choose a threshold τ_2 (e.g., 0.5), where frames with probabilities no less than τ_2 belong to the occurrence interval of E_k

$$\theta_{k,v}^{(n)} \geq \tau_2 \Rightarrow \mathbb{1}\left[T_v \in \left[\hat{T}_k^{s(n)}, \hat{T}_k^{e(n)}\right]\right] = 1, \tag{5}$$

where $\hat{T}_k^{s(n)}$ and $\hat{T}_k^{e(n)}$ represent the start and end offsets of frames in the estimated occurrence interval for event E_k in the time horizon from T_n. Since the events occur in continuous frames, the output frames with $\theta_{k,v}^{(n)} \geq \tau_2$, which may be discontinuous, have to be converted to a continuous frame sequence. The final predicted occurrence interval for Event E_k is

$$\left[\hat{T}_k^{s(n)}, \hat{T}_k^{e(n)}\right] = \left[\min_{T_v \in [1,H]} \{T_v | \theta_{k,v}^{(n)} \geq \tau_2\}, \max_{T_v \in [1,H]} \{T_v | \theta_{k,v}^{(n)} \geq \tau_2\}\right]. \tag{6}$$

The predicted occurrence interval may be affected by false positives. We will report the prediction accuracy in § VI.

As is typical with deep neural architectures, the quality of our resulting estimation can be evaluated experimentally using test datasets. We are interested to move beyond experimental validation of our deep neural architecture and investigate formal quality guarantees for the output of the network. Conformal inference [17] has been successful in providing certain types of probabilistic guarantees, under statistical assumptions, for the output of prediction models. In Sections IV and V, we introduce novel approaches to offer marginal probabilistic guarantees for the predictions of EventHit utilizing a conformal inference approach.

IV. Conformal Event Existence Predictions

In this section, we focus on the classification problem of event existence and propose a method C-CLASSIFY for this problem. We first introduce a well-studied approach called conformal prediction [28], [29], which forms the basis of C-CLASSIFY, and then discuss C-CLASSIFY in detail.

A. Conformal Prediction

Conformal prediction [15] has been widely known in statistics but recently has attained novel interest given its applicability in machine learning for point prediction in classification or regression [16], [30]. The basic principle in conformal predictions is to utilize a reference dataset (calibration set) to determine precise levels of confidence in new predictions. In particular, under the assumptions that the calibration set and the new data we wish to make predictions on are *exchangeable*[4], one can obtain probabilistic guarantees for the new predictions.

[3]More than one event may occur in the time horizon together.

[4]The examples observed from a dataset are exchangeable if for any permutation of the dataset, the joint probability distribution of the permuted dataset is the same as the distribution of the original one. The exchangeability assumption is weaker than the assumption that the random events are independent and identically distributed (i.i.d.), in that all i.i.d. events are exchangeable but not vice versa.

For a given binary classification model, let Δ_{c} be a calibration set, $\Delta_{\text{c}} : \{(x_1, y_1), \ldots, (x_{|\Delta_{\text{c}}|}, y_{|\Delta_{\text{c}}|})\}$, where each x_i is an input feature vector and $y_i \in \{0, 1\}$ is its binary class label. Conformal prediction first constructs a *non-conformity measure* a_i which measures the dissimilarity between a data example x_i and all the data examples in Δ_{c} with respect to the positive label ($y_i = 1$), with higher scores corresponding to higher dissimilarity. Then, for a new data example, the algorithm computes the p-value p_o for this example (x_o, y_o), which is the fraction of data examples in Δ_{c} with higher dissimilarity than x_o,

$$p_o = \frac{|\{i = 1, \ldots, |\Delta_{\text{c}}| : a_o \leq a_i\}|}{|\Delta_{\text{c}}| + 1}.$$

If p_o is small, then $y_o = 1$ is very nonconforming with respect to the past experience. A parameter c is defined as the confidence level, reflecting the required reliability of the prediction. The greater c, the higher the reliability is in our prediction. The quantity $1 - c$ is usually called the significance level. Depending on the confidence level c, a new prediction $\hat{y}_o$ is made,

$$\hat{y}_o = \begin{cases} 1 & p_o \geq 1 - c \\ 0 & p_o < 1 - c \end{cases}. \tag{7}$$

Theorem 4.1: [15] Assuming that $\Delta_{\text{c}} \cup \{(x_o, y_o)\}$ are exchangeable (their distribution is invariant under permutations), the probability of missing the correct positive label when predicting $\hat{y}_o$ as per Equation (7) is guaranteed to be not greater than $1 - c$. In other words, for a data example (x_o, y_o) the following guarantee is valid,

$$P(\hat{y}_o = 0; y_o = 1) \leq 1 - c,$$
$$\text{or} \quad P(\hat{y}_o = 1; y_o = 1) \geq c.$$

This guarantee holds for data sampled in the same way as the conformal dataset, irrespective of the chosen non-conformity measure. The probability is taken over $\Delta_{\text{c}} \cup (x_o, y_o)$ points instead of for a specific data example; namely the guarantee is *marginally* valid as opposed to *conditional*. The estimated labels for different c have the following property:

$$c_1 > c_2 \Rightarrow P(\hat{y}_1 = 1; y_1 = 1, c = c_1) > P(\hat{y}_1 = 1; y_1 = 1, c = c_2).$$

We will utilize principles of conformal predictions to derive probabilistic guarantees utilizing the output scores of EventHit. Moreover, unlike Bayesian approaches for quantifying model output confidence [31], utilizing a conformal framework, there is no need for assumptions on the form of the underlying probability distribution.

B. C-CLASSIFY

EventHit accepts as input covariates $\mathcal{X}_n$ and outputs estimated occurrence scores $b_k^{(n)}$ for each event E_k. Let $\mathbb{C}_k$ be the classification model in EventHit for the prediction of the event occurrence in the sub-network corresponding to event E_k,

$$\mathbb{C}_k(\mathcal{X}_n) = b_k^{(n)}, \quad k = 1, ..., K,$$

Let $\mathcal{D}_{\text{c-calib}}$ be a calibration set that is independently sampled in the same way as the training dataset. We denote $\mathcal{P}_{\text{c-calib}}$ the set of frames utilized to assemble it,

$$\mathcal{D}_{\text{c-calib}} : \{(\mathcal{X}_n, \mathcal{L}_n, \mathcal{T}_n)\}_{n \in \mathcal{P}_{\text{c-calib}}}. \tag{8}$$

$\mathcal{D}_{\text{c-calib}}$ can be a subset of the training dataset $\mathcal{D}$ or have partial overlap with $\mathcal{D}$.

We utilize principles of conformal predictions and propose C-CLASSIFY, an algorithm that calibrates the event prediction scores of EventHit with probabilistic guarantees, effectively eliminating the need to threshold the event prediction output scores. Such estimation takes place for each event independently.

Algorithm 1: C-CLASSIFY

Input: Input Covariates $\mathcal{X}_o$; Conformal Set $\mathcal{D}_{\text{c-calib}}$; Confidence Level c;
Output: Estimated Set of Positive Events $\hat{\mathcal{L}}_o$;

1 **for** $k = 1, ..., K$ **do**
2 Get the score $b_k^{(o)}$ for $\mathcal{X}_o$ through EventHit.
3 $a_o^k = 1 - b_k^{(o)}$.
4 **for** $n \in \mathcal{P}_{\textit{c-calib}}$ **do**
5 Get $b_k^{(n)}$ for $\mathcal{X}_n$ through EventHit.
6 $a_n^k = 1 - b_k^{(n)}$.
7 $p_o^k = \frac{\left|\left\{n \in \mathcal{P}_{\text{c-calib}} : E_k \in \mathcal{L}_n \ \&\ a_o^k \leq a_n^k\right\}\right|}{|\{n \in \mathcal{P}_{\text{c-calib}} : E_k \in \mathcal{L}_n\}| + 1}$.
8 $\hat{\mathcal{L}}_o = \left\{E_k : 1 \leq k \leq K \mid p_o^k \geq 1 - c\right\}$.

C-CLASSIFY is shown in Algorithm 1. Let a_i^k be a non-conformity measure, which is a function returning a real-valued score quantifying the non-feasibility of event E_k occurring in the time horizon given covariates $\mathcal{X}_i$ according to the classifier $\mathbb{C}_k$, with higher scores corresponding to higher non-feasibility. We utilize a simple but commonly employed non-conformity measure[5] [15], [30] namely quantifying the score that signifies that an event does not occur in time horizon, $a_i^k = 1 - b_k^{(i)}$. We calculate the non-conformity of each event for all data in the calibration dataset $\mathcal{D}_{\text{c-calib}}$ (Lines 4-6 in Algorithm 1).

Then, given covariates $\mathcal{X}_o$, a p-value of each event E_k ($k = 1, ..., K$) is calculated with respect to $\mathbb{C}_k$ and the calibration dataset $\mathcal{D}_{\text{c-calib}}$ in the following way (Line 7 in Algorithm 1):

$$p_o^k = \frac{\left|\left\{n \in \mathcal{P}_{\text{c-calib}} : E_k \in \mathcal{L}_n \ \&\ a_o^k \leq a_n^k\right\}\right|}{|\{n \in \mathcal{P}_{\text{c-calib}} : E_k \in \mathcal{L}_n\}| + 1}$$
$$= \frac{\left|\left\{n \in \mathcal{P}_{\text{c-calib}} : E_k \in \mathcal{L}_n \ \&\ (1 - b_k^{(o)}) \leq (1 - b_k^{(n)})\right\}\right|}{|\{n \in \mathcal{P}_{\text{c-calib}} : E_k \in \mathcal{L}_n\}| + 1},$$

where $b_k^{(n)}$ is the estimated score of event E_k for covariates $\mathcal{X}_n$ given by EventHit.

Given a confidence level c, the new estimated set of events that are predicted to occur in time horizon $\hat{\mathcal{L}}_o$ for covariates $\mathcal{X}_o$ is the set of events for which the corresponding p-values are no less than $1 - c$, i.e.,

$$\hat{\mathcal{L}}_o = \left\{E_k : \ p_o^k \geq 1 - c, 1 \leq k \leq K\right\} \tag{9}$$

Following Theorem 4.1, the following theorem holds.

Theorem 4.2: For each event $E_k \in \mathcal{E}$, let $\Delta_c = \{(\mathcal{X}_n, E_k \in \mathcal{L}_n), n \in \mathcal{P}_{\text{c-calib}}\}$ be the records (i.e., triplets) in the calibration dataset, assuming that $\Delta_c \cup (\mathcal{X}_o, E_k \in \mathcal{L}_o)$ are exchangeable[6]; then the following guarantee is valid,

$$P\left(E_k \notin \hat{\mathcal{L}}_o\right) \leq 1 - c. \quad \forall\ E_k \in \mathcal{L}_o.$$

For each event $E_k \in \mathcal{E}$, since $\Delta_c \cup (\mathcal{X}_o, E_k \in \mathcal{L}_o)$ are exchangeable, according to Theorem 4.1, we have $P(E_k \notin$

[5]We stress that Theorem 4.1 holds irrespective of the non-conformity measure utilized. As such any choice of non-conformity measure can be applied to the model proposed in this work. Our entire proposal and ensuing discussion holds verbatim for different choices of non conformity measures.

[6]Since Δ_c reflects the distribution of the results of the classifier $\mathbb{C}_k$ across all frames in the video stream, the test data $(\mathcal{X}_o, E_k \in \mathcal{L}_o)$ that is sampled in the same way as Δ_c has no effect on the overall distribution of $\Delta_c \cup (\mathcal{X}_o, E_k \in \mathcal{L}_o)$, implying that they are exchangeable.

$\hat{\mathcal{L}}_o; E_k \in \mathcal{L}_o) \leq 1-c$. Thus, for any $E_k \in \mathcal{L}_o$, $P\left(E_k \notin \hat{\mathcal{L}}_o\right) \leq 1-c$.

For two confidence levels $c_1 > c_2$, the set $\hat{\mathcal{L}}_{o,c_1}$ estimated under c_1 will include the set $\hat{\mathcal{L}}_{o,c_2}$ estimated under c_2,

$$c_1 > c_2 \Rightarrow \hat{\mathcal{L}}_{o,c_1} \supset \hat{\mathcal{L}}_{o,c_2}. \tag{10}$$

The confidence level c is a user-tunable knob. According to Theorem 4.2, by having the ability to set the confidence level c we can affect the precision and recall of the estimation process. According to Equation (10), for event $E_k \in \mathcal{E}$, setting a higher c is expected to increase the number of E_k that are predicted to occur, leading to a higher recall but a lower precision. To see this, as per Equation (9), a higher value of c decreases the threshold that the corresponding p-value has to surpass for event E_k to be predicted to occur.

Moreover, compared to the approach that determines event occurrence by manipulating thresholds τ_1 as per Equation (4) on the classifier output, C-CLASSIFY disposes of such threshold and manipulates the output via probability semantics. Thus, C-CLASSIFY assists in scaling precision and recall in a tunable manner.

V. Conformal Occurrence Interval Predictions

Equations (5) and (6) in § III enable us to estimate the occurrence intervals $\left[\hat{T}_k^s, \hat{T}_k^e\right]$ for event E_k as produced by EventHit. Once an event is predicted to occur in the time horizon by C-CLASSIFY (i.e., $E_k \in \hat{\mathcal{L}}$), we forward the frames in $\left[\hat{T}_k^s, \hat{T}_k^e\right]$ to the CI to detect the actual event. The accuracy of the estimation of $\hat{T}_k^s$ and $\hat{T}_k^e$ is crucial. Evidently if the two points are not closely aligned with the actual start and end of the event, it may result in missing the event and/or conducting excessive work (processing more frames in the CI than we need to). For this reason, we propose C-REGRESS, which calculates a bound for $\hat{T}_k^s$ and $\hat{T}_k^e$ with specific probabilistic guarantees that can be tuned.

A. Conformal Regression

The principles of conformal predictions apply to regression problems as well. Let $\Delta_\text{r} : \{(x_1, y_1), \ldots, (x_{|\Delta_\text{r}|}, y_{|\Delta_\text{r}|})\}$ be exchangeable pairs of random variables, with x_i being a multi-dimensional vector (d-dimensional feature vectors or covariates) and y_i a response variable. Let $\mu(x) = E[Y|X = x], x \in R^d$ be a regression function. We are interested in predicting y_o for a new vector x_o with no assumptions on μ and the form of the underlying data distribution of Δ_r.

Conformal regression constructs an interval C (a prediction band) based on Δ_r with the property that for a new point (x_o, y_o),

$$P\left(y_o \in C(x_o)\right) \geq \alpha$$

for a user-defined *coverage* level α.

There are multiple ways to construct the prediction band C. We outline below one called split conformal regression [16] which is computationally efficient. We randomly split Δ_r into two equal-sized subsets I_1, I_2. Let $\hat{\mu}$ be a regression function trained on I_1, and

$$r_i = |y_i - \hat{\mu}(x_i)|, \quad i \in I_2,$$

Algorithm 2: C-REGRESS

Input: Input Covariates $\mathcal{X}_o$; Calibration Set $\mathcal{D}_{\text{r-calib}}$; Coverage Level α;
Output: Estimated Occurrence Interval $\hat{\mathcal{T}}_o$

Get output vector $\left[\Theta_1^{(o)}, ..., \Theta_k^{(o)}\right]$ for $\mathcal{X}_o$ through EventHit.
Get $\hat{\mathcal{L}}_o$ for $\mathcal{X}_o$ through C-CLASSIFY.
for $k = 1, ..., K$ **do**
  Get $\left(\hat{T}_k^{s(o)}, \hat{T}_k^{e(o)}\right)$ for $\mathcal{X}_o$ according to $\left[\Theta_1^{(o)}, ..., \Theta_k^{(o)}\right]$.
  Initialize R_k^s and R_k^e as empty sets.
  for $n \in \mathcal{P}_{\textit{r-calib}}$ **do**
    Get $\left(\hat{T}_k^{s(n)}, \hat{T}_k^{e(n)}\right)$ through EventHit w.r.t. $\mathcal{X}_n$.
    if $E_k \in \mathcal{L}_n$ **then**
      $r_n^{s,k} = \left|\hat{T}_k^{s(n)} - T_k^{s(n)}\right|$.
      add $r_n^{s,k}$ into R_k^s.
      $r_n^{e,k} = \left|\hat{T}_k^{e(n)} - T_k^{e(n)}\right|$.
      add $r_n^{e,k}$ into R_k^e.
  Sort and denote the residuals in R_k^s: $r_{(1)}^{s,k} \leq ... \leq r_{(|R_k|)}^{s,k}$.
  Sort and denote the residuals in R_k^e: $r_{(1)}^{e,k} \leq ... \leq r_{(|R_k|)}^{e,k}$.
  $\hat{q}_k^s = r_{(\lceil \alpha(|R_k|)\rceil)}^{s,k}$.
  $\hat{q}_k^e = r_{(\lceil \alpha(|R_k|)\rceil)}^{e,k}$.
$\hat{T}_{k,\text{new}}^{s\ (o)} = \max\{1, \hat{T}_k^{s(o)} - \hat{q}_k^s\}$.
$\hat{T}_{k,\text{new}}^{e\ (o)} = \min\left\{H, \hat{T}_k^{e(o)} + \hat{q}_k^e\right\}$.
$\hat{\mathcal{T}}_o = \left\{\left(\hat{T}_{k,\text{new}}^{s\ (o)}, \hat{T}_{k,\text{new}}^{e\ (o)}\right), \forall\ k : E_k \in \hat{\mathcal{L}}_o\right\}$.

be the fitted residuals for the members of I_2. Let $\hat{q}$ be the $\lceil |\Delta_\text{r}| \cdot \alpha \rceil_\text{th}$ smallest value in $\{r_i : (x_i, y_i) \in I_2\}$, namely the α-quantile of the fitted residuals. For a feature vector x_o, split conformal regression outputs the prediction band $[\hat{\mu}(x_o) - \hat{q}, \hat{\mu}(x_o) + \hat{q}]$, with the following guarantee:

Theorem 5.1: [16] Assuming that $\Delta_\text{r} \cup \{(x_o, y_o)\}$ are exchangeable (their distribution is invariant under permutations), for a user-defined coverage α, we have

$$P(y_o \in [\hat{\mu}(x_o) - \hat{q}, \hat{\mu}(x_o) + \hat{q}]) \geq \alpha.$$

The probability is taken over $\Delta_\text{r} \cup (x_o, y_o)$ and the guarantee is marginally but not conditionally valid, as in the case of conformal classification [16]. The width of this prediction band can be tuned by varying α: larger values of α lead to larger prediction intervals. The prediction bands for different α have the following property:

$$\alpha_1 < \alpha_2 \Rightarrow [\hat{\mu}(x_o) - \hat{q}_{\alpha_1}, \hat{\mu}(x_o) + \hat{q}_{\alpha_1}] \subset [\hat{\mu}(x_o) - \hat{q}_{\alpha_2}, \hat{\mu}(x_o) + \hat{q}_{\alpha_2}].$$

where $\hat{q}_{\alpha_1}$/$\hat{q}_{\alpha_2}$ are the α_1/α_2-quantiles of the fitted residuals.

B. C-REGRESS

We now detail C-REGRESS which is based on the principles of conformal regression. Let $\mathcal{D}_{\text{r-calib}}$ be a calibration set that is independently sampled in the same way as the training dataset, and $\mathcal{P}_{\text{r-calib}}$ be the set of frames utilized to assemble it:

$$\mathcal{D}_{\text{r-calib}} : \{(\mathcal{X}_n, \mathcal{L}_n, \mathcal{T}_n)\}_{n \in \mathcal{P}_{\text{r-calib}}}.$$

Similar to $\mathcal{D}_{\text{c-calib}}$ in Equation (8), $\mathcal{D}_{\text{r-calib}}$ can be a subset of the training dataset $\mathcal{D}$ or have joint records with $\mathcal{D}$.

For an instance $\mathcal{X}_o$, EventHit estimates the occurrence interval (OI) of the events in the time horizon as per Equation (6), $\left[\hat{T}_k^{s(o)},\ \hat{T}_k^{e(o)}\right]$, $\forall\ k = 1, ..., K\ \&\ E_k \in \hat{\mathcal{L}}_o$, where $\hat{\mathcal{L}}_o$ is determined by C-CLASSIFY. C-REGRESS conducts conformal regression on the start and end frame estimates of

the occurrence interval for each event. The entire procedure is depicted in Algorithm 2, which performs the following tasks:

1. Evaluate EventHit on all the calibration data $\mathcal{X}_n : n \in \mathcal{P}_{\text{r-calib}}$ and obtain the estimated occurrence interval of each event $E_k \in \mathcal{E}$ for each data record, $\left(\hat{T}_k^{s(n)}, \hat{T}_k^{e(n)}\right)$ (Line 7).
2. Compute the residuals of both start and end frame estimates of each data record for each event in the calibration set (Line 9),
$$r_n^{s,k} = \left|\hat{T}_k^{s(n)} - T_k^{s(n)}\right|, \quad r_n^{e,k} = \left|\hat{T}_k^{e(n)} - T_k^{e(n)}\right|, \quad \forall\, n \in \mathcal{P}_{\text{r-calib}}\ \forall\, E_k \in \mathcal{L}_n.$$
Let $|R_k|$ be the number of records in the calibration set with event E_k in its time horizon,
$$|R_k| = |\{n \in \mathcal{P}_{\text{r-calib}} : E_k \in \mathcal{L}_n\}|.$$
3. For each event $E_k \in \mathcal{L}_n$, we sort the residuals for both start and end frame estimates (Lines 13,14),
$$r_{(1)}^{s,k} \le r_{(2)}^{s,k} \le ... \le r_{(|R_k|)}^{s,k}, \quad r_{(1)}^{e,k} \le r_{(2)}^{e,k} \le ... \le r_{(|R_k|)}^{e,k}.$$
4. Let $\hat{q}_k^s$ and $\hat{q}_k^e$ be the α-th quantile of the residuals of both start and end frame estimations for each event $E_k \in \mathcal{L}_n$ (Lines 15,16),
$$\hat{q}_k^s = r_{(\lceil \alpha |R_k| \rceil)}^{s,k}, \quad \hat{q}_k^e = r_{(\lceil \alpha |R_k| \rceil)}^{e,k}.$$
5. C-REGRESS adjusts the start point of the estimated occurrence interval to $\hat{T}_{k,\text{new}}^{s\ (o)} = \max\{1,\ \hat{T}_k^{s(o)} - \hat{q}_k^s\}$, and the end point to $\hat{T}_{k,\text{new}}^{e\ (o)} = \min\{H,\ \hat{T}_k^{e(o)} + \hat{q}_k^e\}$. Thus, the estimated set of occurrence intervals becomes
$$\hat{\mathcal{T}}_o = \left\{\left(\max\left\{1, \hat{T}_k^{s(o)} - \hat{q}_k^s\right\}, \min\left\{H, \hat{T}_k^{e(o)} + \hat{q}_k^e\right\}\right), \forall\ k : E_k \in \hat{\mathcal{L}}_o\right\}. \tag{11}$$

Following Theorem 5.1, we can show:

Theorem 5.2: For each event $E_k \in \mathcal{L}_o \cap \hat{\mathcal{L}}_o$, assume that $(\mathcal{X}_o,\ T_k^{s(o)}) \cup (\mathcal{X}_n,\ T_k^{s(n)})$, $n \in \mathcal{P}_{\text{r-calib}}$, and the data for E_k used to train EventHit are exchangeable, and $(\mathcal{X}_o,\ T_k^{e(o)}) \cup (\mathcal{X}_n,\ T_k^{e(n)})$, $n \in \mathcal{P}_{\text{r-calib}}$, and the data for E_k used to train EventHit are exchangeable; then the following holds,

$$P\left(T_k^{s(o)} \in [\hat{T}_k^{s(o)} - \hat{q}_k^s, \hat{T}_k^{s(o)} + \hat{q}_k^s]\right) \ge \alpha. \quad \forall\ E_k \in \mathcal{L}_o \cap \hat{\mathcal{L}}_o;$$

$$P\left(T_k^{e(o)} \in [\hat{T}_k^{e(o)} - \hat{q}_k^e, \hat{T}_k^{e(o)} + \hat{q}_k^e]\right) \ge \alpha. \quad \forall\ E_k \in \mathcal{L}_o \cap \hat{\mathcal{L}}_o.$$

Since the new start point of the estimated occurrence interval $\hat{T}_{k,\text{new}}^{s\ (o)} \le \hat{T}_k^{s(o)}$, the estimated start frame is made earlier in time. Similarly, since $\hat{T}_{k,\text{new}}^{e\ (o)} \ge \hat{T}_k^{e(o)}$, the estimated end frame is moved forward in time. According to Theorem 5.2, setting larger values of α leads to larger estimated occurrence intervals, and as a result we are more likely not to miss events; however, the flip side of this is that more irrelevant frames are likely to be processed by CI, resulting in more waste of resources. Thus, similar to c proposed in § IV-A, by varying α, we obtain a tunable trade-off between recall and cost savings for each event $E_k \in \mathcal{E}$, which we will explore further in § VI.

VI. Experimental Evaluation

A. Datasets

1. VIRAT [32], a realistic and natural dataset for video surveillance domains with diverse scenes and a variety of event categories. We select 6 representative event types from VIRAT, as shown in Table I along with their occurrence and duration information. As detailed in the preceding sections, our work is focused on utilizing a CI in a subscription mode as is possesses the latest and most advanced models for the events of interest. As such, the CI will provide the accurate models for detecting the events in VIRAT.
2. THUMOS [33], [34], a large-scale video dataset for recognizing and localizing wide ranges of human actions. We select 3 representative action types from it, as predictable events, which are listed in Table I.
3. Breakfast [35], an action recognition dataset consisting of several cooking activities. Each video records the entire process of completing an activity that is completed by a set of action units. We selected 3 action units randomly as the target events in the experiments.

TABLE I: Events of interest in VIRAT and THUMOS.

Event Types in VIRAT	Occurrences	Duration	
		Avg.	Std.
E_1: Person Opening a Vehicle	54	70.8	15.4
E_2: Person Closing a Vehicle	57	62.0	11.9
E_3: Person Unloading an Object from a Vehicle	56	86.6	25.0
E_4: Person getting into a Vehicle	93	145.1	35.1
E_5: Person getting out of a Vehicle	162	193.7	158.8
E_6: Person carrying an object	165	571.2	176.4
Event Types in THUMOS	**Occurrences**	**Duration**	
		Avg.	Std.
E_7: Volleyball Spiking	80	99.3	40.1
E_8: Diving	74	91.2	35.4
E_9: Soccer Penalty	48	92.8	25.9
Event Types in Breakfast	**Occurrences**	**Duration**	
		Avg.	Std.
E_{10}: Cut Fruit	132	114.0	48.8
E_{11}: Put fruit to Bowl	121	97.2	107.5
E_{12}: Put Egg to Plate	95	240.2	153.8

We evaluate our proposed approaches and the baseline algorithms on several prediction tasks, each of which consists of a subset of the events in Table I. The specific tasks are listed in Table II. For the tasks consisting of more than one event, the occurrences of all events are predicted. For each event in VIRAT, we select descriptive features utilizing the annotation provided in the dataset. For example, for event E_1, we have features such as an indicator of the presence/absence of moving cars and a value for the average distance between the cars and the persons in a frame. For each event in THUMOS, we select descriptive features utilizing the object detection outputs produced by the detection models on the videos of the dataset. For each event in Breakfast, we select descriptive features utilizing the object detection outputs produced by the detection models and the annotation of action units appearing in the collection window provided in the dataset. We utilize light-weight widely used models (such as YOLOv3 or Faster R-CNN [36]) for feature selection purposes. All prediction methods may benefit from more task-specific feature engineering. We treat it as an orthogonal issue as standard feature selection methods will apply [25]. Approaches such as frame sampling [37] or difference detector [38] can speed up video processing and can be readily applied in our approach. However they are orthogonal to the methods proposed in this paper.

In the experimental evaluation, we assume that CI can provide an accurate model for predicting the set of events corresponding to each task. The goal is to decide whether the

time horizon (or part of it) should be relayed to CI for further processing. In general, more time horizons being relayed to CI means higher recall and more resource over-utilization. Our experimental evaluation is organized as follows. We introduce the evaluation measures in § VI.C and compare the overall performance of our proposals with other algorithms (listed in § VI.B) in § VI.D, and then conduct ablation studies to evaluate the contribution of conformal predictions in § VI.E, which is followed by studies on the sensitivity of the proposed methods to hyper-parameters in § VI.F. Moreover, we conduct a case study to demonstrate the monetary savings of our proposals in § VI.G. Finally, we report other details of the proposals such as training/inference time and memory used in § VI.H.

B. Algorithms Compared

1. EHO: It utilizes the output of EventHit only to determine the existence of events via Equation (4) and estimate the occurrence intervals via Equation (5) for events that are predicted to occur in the time horizon. The event prediction threshold τ_1 and occurrence interval prediction threshold τ_2 are both set to 0.5.
2. EHC: It utilizes C-CLASSIFY to determine the existence of events of interest (Equation (9)) but keeps using the occurrence interval estimation made by EventHit (Equation (5)). We omit the threshold τ_1 and set τ_2 to 0.5.
3. EHR: It utilizes the output of EventHit to determine the existence of events (Equation (4)) and produce the occurrence intervals for events that are predicted to occur in the time horizon by C-REGRESS (Equation (11)). The thresholds τ_1 and τ_2 are both set to 0.5.
4. EHCR: It utilizes C-CLASSIFY to determine the existence of events of interest (Equation (9)) and C-REGRESS to estimate the occurrence intervals (Equation (11) with $\tau_2 = 0.5$) according to the output of EventHit.
5. OPT: It is the theoretically optimal approach which has full knowledge of all true event intervals relevant to the prediction task and applies the CI onto the frames that true events occur only.
6. BF: The Brute-Force approach that applies the CI onto every single frame in the video.
7. COX: We consider a strategy based on Cox's proportional hazard model [39], a survival regression model in statistics. It builds the model on the covariates and estimates the survival probability. Given a threshold τ_{cox}, we can iterate over the frames in the time horizon to identify the first frame (that will be considered as the start point) whose survival probability is greater than τ_{cox}, assuming that the events will occur from this frame to the end of the time horizon[7].
8. VQS: We also adapt a well-known video query system, BlazeIt [40] for this problem, which utilizes specialized models to filter out video frames that do not satisfy object-based predicates. Given a threshold τ_{vqs}, VQS can relay the time horizon in which the number of frames containing target object types (associated with the events of interest) exceeds the threshold τ_{vqs} to the CI for further processing; it will filter out those time horizons below the threshold τ_{vqs}.
9. APP-VAE: [41] proposed a point-process-based approach for action appearance and arrival time prediction. It can encode past asynchronous time action sequences and make predictions on which actions will occur and when they will happen using a generative model. Using its prediction of action occurrence and arrival time, we can relay only the range of frames that may include target events to CI for further processing. The settings follow [41].

[7]We disregard the end point, as the Cox model can only regress one variable.

TABLE II: Tasks.

Tasks	Events of Interest	Tasks	Events of Interest
TA_1	$\mathcal{E} = \{E_1\}$	TA_9	$\mathcal{E} = \{E_1, E_5, E_6\}$
TA_2	$\mathcal{E} = \{E_2\}$	TA_{10}	$\mathcal{E} = \{E_7\}$
TA_3	$\mathcal{E} = \{E_3\}$	TA_{11}	$\mathcal{E} = \{E_8\}$
TA_4	$\mathcal{E} = \{E_4\}$	TA_{12}	$\mathcal{E} = \{E_9\}$
TA_5	$\mathcal{E} = \{E_5\}$	TA_{13}	$\mathcal{E} = \{E_{10}\}$
TA_6	$\mathcal{E} = \{E_6\}$	TA_{14}	$\mathcal{E} = \{E_{11}\}$
TA_7	$\mathcal{E} = \{E_1, E_5\}$	TA_{15}	$\mathcal{E} = \{E_{11}, E_{12}\}$
TA_8	$\mathcal{E} = \{E_5, E_6\}$	TA_{16}	$\mathcal{E} = \{E_{10}, E_{12}\}$

All algorithms were implemented in Python and run on a Linux server with Intel Xeon Gold 6244 3.60GHz CPU and 64GB memory and an NVIDIA RTX TITAN Xp GPU.

C. Measures

1) Measuring End-to-End Performance.

We measure both the accuracy and the resulting cost savings of the algorithms compared. For the end-to-end accuracy, we are concerned with measuring the recall rate [42], i.e., how many frames belonging to an actual event occurrence are correctly predicted by an algorithm and sent to the CI for processing. To this end, we define η_n^k for event E_k and the n-th record in the dataset as the fraction of frames of the actual event occurrence interval that is predicted by the estimate of a given algorithm:

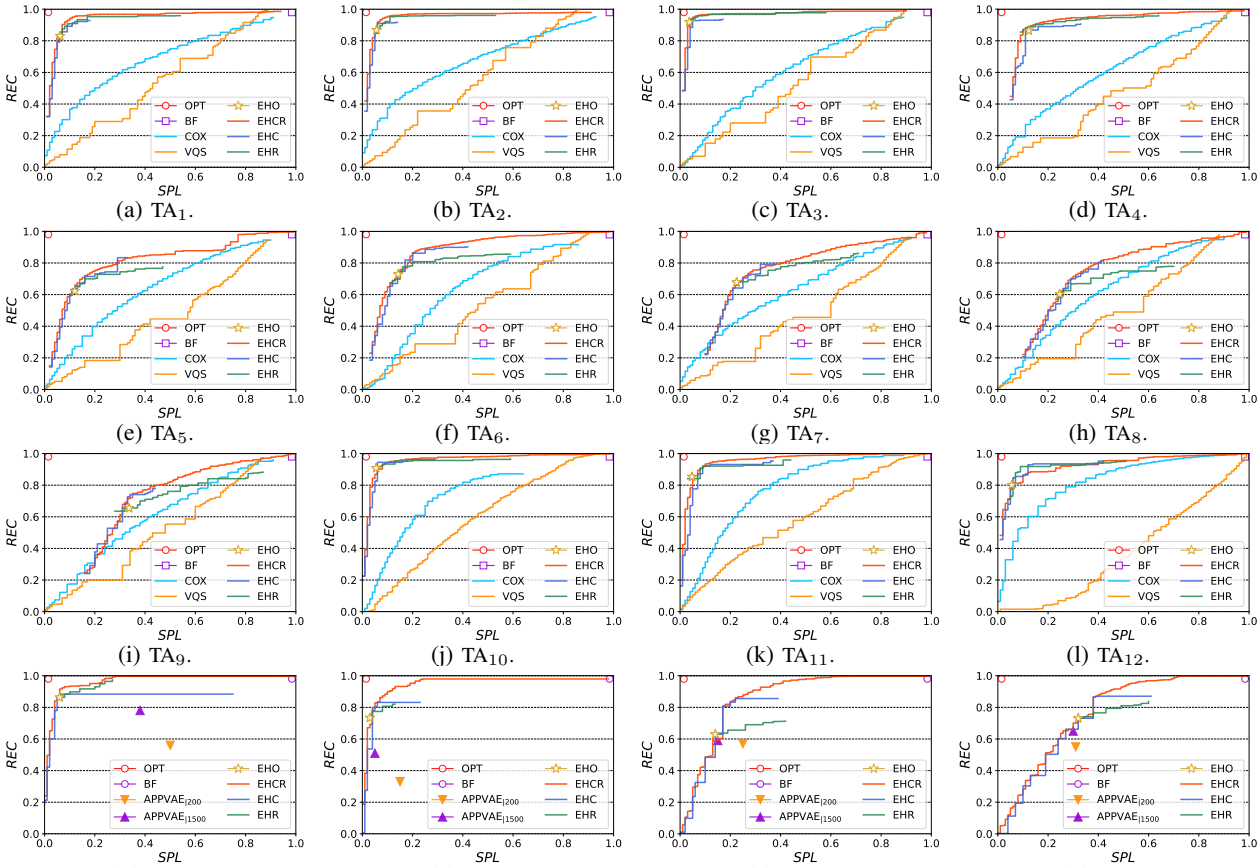
$$\eta_n^k = \frac{\left|\text{range}\left(\hat{T}_k^{s(n)}, \hat{T}_k^{e(n)}\right) \cap \text{range}\left(T_k^{s(n)}, T_k^{e(n)}\right)\right|}{\left|\text{range}\left(T_k^{s(n)}, T_k^{e(n)}\right)\right|},$$

where $\text{range}(T_1, T_2)$ represents the set of frames from T_1 to T_2 ($T_1 < T_2$). In a sense, η_n^k can be thought of as the frame-level recall for event E_k and the n-th record. If an event E_k is predicted not to occur in the next time horizon (i.e., $E_k \notin \hat{\mathcal{L}}_n$), we set $\text{range}(\hat{T}_k^{s(n)}, \hat{T}_k^{e(n)}) = \phi$.

To measure the end-to-end accuracy of an algorithm, we define *REC*, which is the average value of η_n^k for all the events across all records,

$$REC = \frac{\sum_{n \in \mathcal{P}_{\text{test}}} \sum_{E_k \in \mathcal{E}} \mathbb{1}\left[E_k \in \mathcal{L}_n\right] \cdot \eta_n^k}{\sum_{n \in \mathcal{P}_{\text{test}}} \sum_{E_k \in \mathcal{E}} \mathbb{1}\left[E_k \in \mathcal{L}_n\right]}. \tag{12}$$

To measure the end-to-end cost savings, we define the *spillage* of an algorithm, denoted by *SPL*, that computes the following quantity: out of all the frames that do not belong to an event, what is the proportion of frames that are sent to the CI for processing? In a sense, *SPL* can be thought of as the frame-level false positive rate [42]. Since the processing of such frames would not return any positive identification of events, it is wasteful to send them to the CI. Therefore, the lower the spillage, the better. Ideally, this value should be 0, i.e., no resource is wasted on processing those irrelevant



(a) TA_1. (b) TA_2. (c) TA_3. (d) TA_4.
(e) TA_5. (f) TA_6. (g) TA_7. (h) TA_8.
(i) TA_9. (j) TA_{10}. (k) TA_{11}. (l) TA_{12}.
(m) TA_{13}. (n) TA_{14}. (o) TA_{15}. (p) TA_{16}.

Fig. 4: *REC-SPL* curves.

frames, which is the case for the optimal algorithm OPT. On the other extreme, spillage is 1 for the brute-force algorithm BF. More precisely, spillage is defined as

$$SPL = \frac{1}{|\mathcal{P}_{\text{test}}| \cdot |\mathcal{E}|} \sum_{n \in \mathcal{P}_{\text{test}}} \sum_{E_k \in \mathcal{E}} \left[\mathbb{1}\left[E_k \in \mathcal{L}_n \ \&\ E_k \in \hat{\mathcal{L}}_n\right] \frac{\left|\text{range}\left(\hat{T}_k^{e(n)}, \hat{T}_k^{s(n)}\right) \setminus \text{range}\left(T_k^{e(n)}, T_k^{s(n)}\right)\right|}{H - \left|\text{range}\left(T_k^{e(n)}, T_k^{s(n)}\right)\right|} + \mathbb{1}\left[E_k \notin \mathcal{L}_n \ \&\ E_k \in \hat{\mathcal{L}}_n\right] \frac{\left|\text{range}\left(\hat{T}_k^{e(n)}, \hat{T}_k^{s(n)}\right)\right|}{H} \right]. \tag{13}$$

To evaluate the performance of the algorithms and each of their stages (e.g., feature extraction), we use *FPS*, expressing the average number of frames processed per second (FPS) by the algorithms or by their corresponding stage.

2) Measuring Individual Components.

We now describe the measures used to evaluate the effectiveness of the individual components in our proposed approach. In this paper, the problem of predicting when the events of interest occur consists of two successive tasks: the *existence prediction* (whether an event will occur in the horizon), and the *occurrence interval prediction* (what is the occurrence interval given that the events are predicted to occur). The existence prediction task can be considered as a classification problem and optimized using C-CLASSIFY. For event $E_k \in \mathcal{E}$, we use *positive* (or *negative*) *records* to refer to the records whose time horizons are predicted to (or not to) experience the events of interest, i.e., $E_k \in \hat{\mathcal{L}}_n$ (or $E_k \notin \hat{\mathcal{L}}_n$). We define REC_{c} as the recall of the existence prediction task, calculated by:

$$REC_{\text{c}} = \frac{\sum_{n \in \mathcal{P}_{\text{test}}} \sum_{E_k \in \mathcal{E}} \mathbb{1}\left[E_k \in \mathcal{L}_n \ \&\ E_k \in \hat{\mathcal{L}}_n\right]}{\sum_{n \in \mathcal{P}_{\text{test}}} \sum_{E_k \in \mathcal{E}} \mathbb{1}\left[E_k \in \mathcal{L}_n\right]}.$$

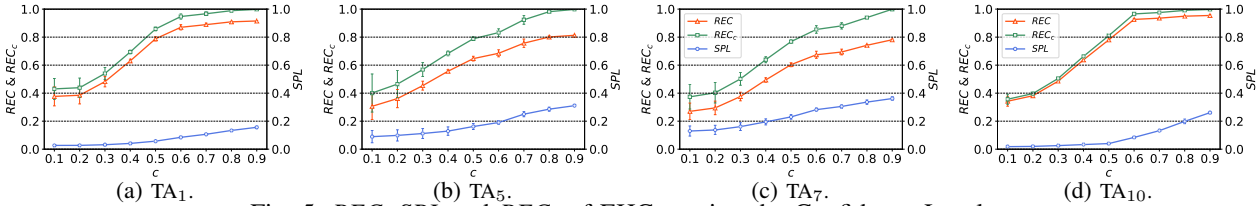
The records with events predicted to occur in the horizon are processed by C-REGRESS to estimate the occurrence interval. To measure the accuracy of this task, we define REC_{r}, the average value of η_n^k for all the events over all those records with events predicted to occur:

$$REC_{\text{r}} = \frac{\sum_{n \in \mathcal{P}_{\text{test}}} \sum_{E_k \in \mathcal{E}} \mathbb{1}\left[E_k \in \mathcal{L}_n \ \&\ E_k \in \hat{\mathcal{L}}_n\right] \cdot \eta_n^k}{\sum_{n \in \mathcal{P}_{\text{test}}} \sum_{E_k \in \mathcal{E}} \mathbb{1}\left[E_k \in \mathcal{L}_n \ \&\ E_k \in \hat{\mathcal{L}}_n\right]}.$$

D. Experimental Results

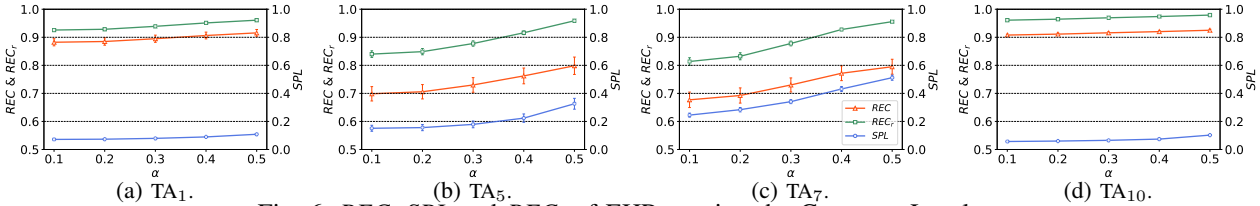
We take the average of 10 independent trials for each combination of task and algorithm compared and report the results. Unless stated otherwise, we set the size of the collection window M=25 and the length of the time horizon H=500 for events in VIRAT, set M=10 and H=200 for THUMOS, and set M=50 and H=500 for Breakfast.

Figure 4 presents the performance of all algorithms compared. The REC of approaches OPT and BF are both 1 for all tasks as they always send all frames with events to the CI. Also, Since BF applies CI onto all frames blindly, its spillage SPL is always 1. By varying the confidence level c and the coverage level α, we obtain the REC-SPL curves for approaches EHC,



(a) TA_1. (b) TA_5. (c) TA_7. (d) TA_{10}.

Fig. 5: *REC*, *SPL* and REC_c of EHC varying the Confidence Level c.



(a) TA_1. (b) TA_5. (c) TA_7. (d) TA_{10}.

Fig. 6: *REC*, *SPL* and REC_r of EHR varying the Converge Level α.

EHR and $EHCR$. Since the outcomes of EHO and APP-VAE cannot be adjusted explicitly via tunable knobs (such as c and α), they are represented as a single point respectively in the plots. While COX and VQS lack explicit tunable knobs as well, we can nevertheless obtain the REC-SPL curves for them by setting different thresholds. The closer the curves or points of an algorithm are to the upper left corner, the better the performance, as this indicates that at the same level of spillage SPL, this algorithm is able to achieve a higher recall REC, and vice versa.

APP-VAE requires an excessively large collection window M to obtain an accurate prediction. In contrast to other approaches, we give APP-VAE larger M values: M=200 and M=1500, represented by APP-VAE$_{|200}$ and APP-VAE$_{|1500}$ respectively. However, a larger M requires more time in the feature extraction phase to identify the action units appearing in the collection window[8]. We did not run APP-VAE on VIRAT and THUMOS since their event occurrences are too sparse and the M required by APP-VAE is insurmountable. The continuous nature of the actions in the videos of the dataset Breakfast makes it more compact for APP-VAE.

It can be observed from Figure 4 that EHO significantly outperforms COX and VQS on VIRAT and THUMOS[9], as well as APP-VAE$_{|200}$ on Breakfast. APP-VAE$_{|1500}$ is worse than EHO on tasks TA_{13} and TA_{14} (i.e., one-event cases), and is comparable to EHO on tasks TA_{15} and TA_{16} (i.e., multi-event cases). For some tasks (such as TA_1, TA_2, TA_{10} and TA_{13} in Figures 4a, 4b, 4j and 4m) EHO has a high REC (> 0.9) and a low SPL (< 0.1). However, with the same network structure and parameters, its performance is not as good on some other tasks (such as TA_5 and TA_6 in Figures 4e and 4f). To understand the performance trends, we divide the events into two groups based on the statistics presented in Table I.

[8]For APP-VAE [41], its encoder and generation model take around 0.1s for inference. However, APP-VAE$_{200}$ requires around 7 seconds for feature extraction since the speed of common action detection models is approximately 25 frames per second [43], [44]. For APP-VAE$_{|1500}$, its feature extraction will require 1 minute, which drastically reduces its competitiveness.

[9]For better presentation, we do not include the curves of COX and VQS in Figures 4m-4p, which exhibit similar performance trends on Breakfast as other datasets.

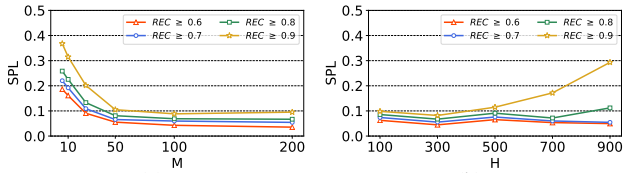
Group 1: Events with short average occurrence duration and small standard deviation, i.e., E_1, E_2, E_3, E_4, E_7, E_8, E_9 and E_{10}.

Group 2: Events with long average occurrence duration or large standard deviation, i.e., E_5, E_6 E_{11} and E_{12}.

By comparing the figures of tasks involving Group 1 events with those of tasks involving Group 2, we observe that EHO performs better (i.e., higher REC and lower SPL) on the former category of tasks. The reason is that on one hand, the higher variation in the duration of the occurrences (e.g., Std.=158.8 for E_5) makes it more difficult to estimate the intervals and thus leads to lower REC (in Figure 4e); on the other hand, SPL tends to be higher for events with longer occurrence duration (e.g., Avg.=571.2 for E_6), because the average number of frames in the estimated occurrence intervals over all records, including those false positive records (i.e., $E_k \notin \mathcal{L}_n$ & $E_k \in \hat{\mathcal{L}}_n$), are larger (e.g., 285 for E_6 vs. 103 for E_1), resulting in higher cost due to false positive records. Moreover, the events with longer occurrence occupy a larger proportion of the time horizon, leaving fewer frames outside the estimated occurrence intervals.

In addition to tasks containing a single event, we select some representative tasks involving multiple events, shown in Figures 4g, 4h, 4i, 4o, and 4p. It can be observed that the performance of EHO on TA_7 and TA_8 is worse than that on TA_1 and TA_6 respectively, but comparable to that on TA_5. The performance on TA_9 is worse than on the former two. It is also interesting to observe that for the combination of events in a task, the overall performance is bound by the event with the worst performance.

C-CLASSIFY or C-REGRESS show considerable flexibility in trading SPL for higher REC, especially when REC of EHO is relatively low, such as on tasks TA_5 or TA_7. In Figures 4e and 4g, we observe that the curves of EHC and EHR can reach higher REC than EHO. However, neither EHC nor EHR can reach a REC value close to 1. On the other hand, EHCR, which combines C-CLASSIFY and C-REGRESS, is able to achieve any required level of REC. In each of the plots, the curves of EHCR can always reach the maximum REC, albeit at the expense of higher SPL. In comparison to the tasks



(a) (b)

Fig. 7: *SPL* of EHCR under different *REC* levels varying the size of the collection window M (Left) and the size of the time horizon H (Right) on task TA_1.

involving Group 1 events (e.g., Figures 4a, 4b, 4c and 4d), EHCR incurs a higher SPL to obtain the same level of REC on tasks involving Group 2 events (e.g., Figures 4e and 4f). Similarly, in comparison to the tasks involving fewer event types (e.g., Figures 4a and 4e), EHCR requires a higher SPL to obtain the same level of REC on tasks involving more event types (e.g., Figures 4g, 4h and 4i). This is as expected since having more complexity in the task makes predictions more challenging.

E. Evaluation of Conformal Prediction

We next conduct experiments using EHC and EHR to analyze the effectiveness of C-CLASSIFY and C-REGRESS.

As per § IV, C-CLASSIFY tunes the output of EventHit via conformal prediction and delivers the level of recall (REC_c) for a given confidence level c. We thus vary c and evaluate EHC with respect to REC, SPL, and REC_c on four representative tasks. The results are shown in Figure 5. Consistent across all four plots, a larger c leads to a higher REC at the expense of a higher SPL, because a larger c increases the chances a time horizon to be predicted as positive. As REC is influenced by both the recall of the event existence classification and the recall of the occurrence interval prediction, we additionally plot the curves of REC_c with varying c. As c approaches 1, almost all the records are predicted to be positive; thus REC_c reaches 1 for all the tasks. However, REC cannot reach 1, due to errors in estimating event occurrence intervals.

We next explore the effect of C-REGRESS, which aims to manipulate the start and end points of the estimated occurrence intervals at the specified coverage level α. Figure 6 presents the performance of EHR with respect to REC, SPL and REC_r on four representative tasks, with varying coverage level α. As discussed in § V, a larger α leads to larger estimated occurrence intervals, resulting in a higher REC and a higher SPL. However, the effect of α differs across the tasks. For some tasks (such as TA_1 and TA_{10}), on which EHO already has a high REC_r, increasing α results in limited improvement (Figure 6a and 6d); for tasks whose value of REC_r resulting by EHO is low (such as TA_5 and TA_7), α can significantly improve their REC_r (Figures 6b and 6c). This observation is applicable to all tasks. Among all the event types, although their resulting REC_r by EHO varies, their REC_r can reach 0.95 when $\alpha = 0.5$, with a small increase of SPL (≤ 0.2) compared with EHO.

In our experiments, c demonstrates a greater impact than α. For some tasks whose events of interest belong in Group 1, since REC_r resulting from EHO is already very high, α has a low impact on REC; thus, in these cases, it is better to set a small α. In contrast, for tasks with events in Group 2, whose occurrence interval prediction with EHO is inaccurate, increasing α can have a remarkable improvement in REC. Thus, α can be used to further increase REC when tuning c alone cannot reach the ideal REC.

F. Sensitivity to Hyper-Parameters

In Figure 7a, we vary M and present the different *SPL* values required to reach different *REC* levels on task TA_1. In general, a larger collection window (larger M) offers more temporal context when making predictions and thus leads to better performance. As shown in Figure 7a, the performance can be considerably enhanced by raising M until M reaches 50; beyond that, the benefit of continuing increasing M exhibits diminishing returns. Since feature extraction from frames incurs overhead, it is not advisable to increase M arbitrarily. As such, M=50 is a good choice for this dataset. We would like to note that the optimal choice of M may vary from one dataset to another and can be chosen experimentally by hyper-parameter search [23], [24].

Figure 7b illustrates the *SPL* value required to reach different levels of *REC* levels on task TA_1 with varying H. As can be observed from Figure 7b, the effect of H on EHCR is insignificant at low *REC* values such as for the cases of *REC*$\geq$0.6 and *REC*$\geq$0.7. For higher values of *REC* (as in the case of *REC*$\geq$0.8 and *REC*$\geq$0.9), with increasing H, EHCR has an impact on *SPL*, since for a specific event, the fraction (percentage) of the event occurrence interval to the time horizon decreases (e.g., 51% for H=100 and 5.3% for H=900), making estimation of occurrence interval more difficult. The cost of achieving the desired higher *REC* levels is thus higher.

G. Case Study on Monetary Cost

To realize things concrete, we demonstrate the performance of EHCR considering the task TA_1, utilizing the pricing of Amazon Rekognition[10], that is US $0.001 per frame. Figure 8 presents *REC* versus the associated Expense ($) of several approaches. OPT represents the expense of processing only the frames in the true occurrence intervals. BF is the expense associated with processing all frames in all records. As illustrated in Figure 8, EHCR can achieve approximately 100% *REC* with less than a fifth of the expense associated with BF, which is significantly less than the expense of COX for this *REC*. Similar results hold for the other types of tasks and events and are omitted for brevity.

H. Details of Resource Utilization

Figure 9 presents *REC* versus *FPS* for EHCR, COX and VQS for tasks TA_{10} and TA_{11}. The *FPS* of EHCR includes the inference overhead of the feature extraction model (e.g., YOLOv3 [45]), EventHit, and the event detection model (e.g., I3D [43]) provided by the CI. Similarly, for COX we account for feature extraction, the Cox model overhead and the CI processing; for VQS we include the specialized run-time

[10]https://aws.amazon.com/rekognition/pricing/

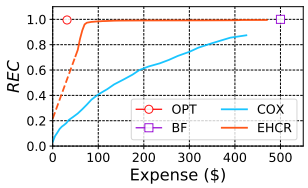
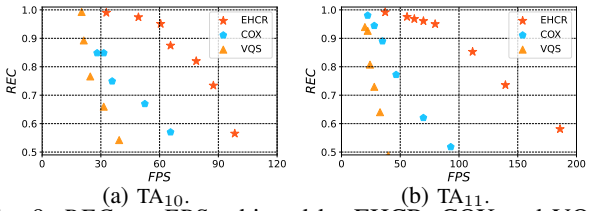


Fig. 8: *REC*-Expense curve on task TA_1.



(a) TA_{10}. (b) TA_{11}.

Fig. 9: *REC* vs. *FPS* achieved by EHCR, COX and VQS.

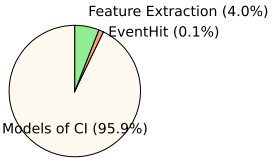


Fig. 10: Proportion of the time overhead on each stage of EHCR.

(inference) model overhead [38], [40] and the CI processing. The graph demonstrates a clear trade-off between *REC* and FPS, and EHCR clearly dominates this trade-off. As we relax *REC*, the FPS of EHCR surpasses that of COX and VQS significantly. For example, for TA_{11}, when REC=0.9, where COX and VQS can only achieve FPS less than 40, EHCR can achieve FPS greater than 100; when REC=0.7, where COX and VQS can only achieve FPS around 50, EHCR can achieve FPS greater than 150. Similar results hold for the other types of tasks and events used in this study. In addition, with the batch size set to 128, the training time of EventHit is usually less than 1 hour and approximately 150MB of GPU memory is required for both training and inference. Figure 10 presents the proportion of the time spent on each of the three stages of EHCR throughout the entire process (for TA_{10} and REC=0.9). As can be observed from the figure, time spent on CI is dominant, which is one of the reasons (along with monetary cost) why it is desirable to reduce CI invocation (which is the main objective of EHCR).

VII. Related Work

Video analytics utilizing deep learning models as primitives is an increasing area of research interest in the database community. Several recent works present query processing frameworks utilizing frame content (objects, spatial locations in frame, etc) as first class citizens to query processing [46]–[50]. For example, NoScope and BlazeIt [38], [40] utilize special purpose-built neural networks (NNs) to detect objects accelerating queries via inference-optimized model search. Some of the query processing frameworks, such as BlazeIt [40] and SVQ [51], [52], have similarities with our model EHCR in that both aim to reduce the overhead of complex detection models. However, video querying frameworks cannot be directly applied to the problems proposed in this paper as they lack the ability to make predictions. Such frameworks have to apply lightweight models on *all* frames instead of predicting the event occurrence intervals and skipping unnecessary frames. The superiority of our proposal over the adaptation of such frameworks was clearly demonstrated by our experiments.

Survival analysis is a collection of statistical techniques that help predict the time until an event occurs [53]. These methods were initially used to predict the time of survival for patients under different treatments, hence the name "survival analysis". For the same reason, the "time until an event occurs" is also called survival time in that literature. Survival analysis has been utilized in the past in the context of data management, such as [18], [54]. Past applications have mainly utilized parametric models of survival analysis as opposed to deep learning. Although some recent research utilized neural networks [54] to tackle the survival analysis problem, EventHit is novel in that it is able to simultaneously predict *if* and *when* (a range) an event will occur.

Predicting if and when events occur in a video frame sequence is a relatively new and emerging topic, such as [55] that proposes a model for assessing future moments of the action of interest. In contrast, our proposal offers a lightweight mechanism to provide tunable statistical guarantees for predictions. We stress that the conformal event existence prediction and conformal occurrence interval prediction algorithms proposed in this paper are applicable to any models capable of predicting the existence (and probability) of events as well as their occurrence intervals. Thus our approach can be wrapped around any existing or future model capable of predictions in diverse problem settings, offering tunable knobs with statistical guarantees. In addition, several works address aspects of video predictions [56]–[60]. Although related, these problems are very different from the focus of our work.

VIII. Conclusions and Future Work

Despite the convenience and processing power offered by cloud-based serving of deep models, the cost of using such services for processing video streams can grow rapidly if the streams are sent to the CI unabridged. In this paper, we consider the problem of marshalling the model inference requests, and propose a framework called EventHit to decide which segments of the video are to be sent to the CI utilizing predictions on event occurrence in a given time horizon. A salient contribution of our work is that we offer two optimizations of EventHit, through novel adoption of the conformal prediction technique. Such optimizations provide tunable knobs for the users to have fine control over the trade-off between accuracy and cost with probabilistic guarantees, thus making the framework highly versatile to serve a wide variety of applications. Extensive evaluation on real datasets has confirmed the superiority of the proposed approach over baseline methods in that it offers significant cost savings and unprecedented control over the accuracy/cost trade-off in cloud-based model serving.

In our study, we have assumed that the occurrence of each type of event follows a stationary underlying distribution. For future work, it would be interesting to investigate how to detect and adapt to changes in the occurrence distribution over time. Another possible direction is to study further optimizations to our proposal that would allow it to meet certain fairness criteria [61] when they are present.

REFERENCES

[1] I. J. Goodfellow, Y. Bengio, and A. C. Courville, *Deep Learning*, ser. Adaptive computation and machine learning. MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org/

[2] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, 2017. [Online]. Available: https://doi.org/10.1109/TPAMI.2016.2577031

[3] H. Zhang, Y. Zhang, B. Zhong, Q. Lei, L. Yang, J. Du, and D. Chen, "A comprehensive survey of vision-based human action recognition methods," *Sensors*, vol. 19, no. 5, p. 1005, 2019. [Online]. Available: https://doi.org/10.3390/s19051005

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017. [Online]. Available: http://doi.acm.org/10.1145/3065386

[5] C. M. Inc, "Meraki mx cloud managed smart cameras," 2021, https://meraki.cisco.com/products/smart-cameras/.

[6] A. R. D. Guide, "What is amazon rekognition," 2021. [Online]. Available: https://docs.aws.amazon.com/rekognition/latest/dg/what-is.html

[7] P. Farley and D. Coulter, "What is custom vision?" 2021. [Online]. Available: https://docs.microsoft.com/en-us/azure/cognitive-services/custom-vision-service/overview

[8] T. Cheng, "Introduction to google cloud vision," 2021. [Online]. Available: https://nanonets.com/blog/google-cloud-vision/

[9] IBM, "What is watson visual recognition?" 2021. [Online]. Available: https://www.ibm.com/dk-en/cloud/watson-visual-recognition

[10] A. C. Community, "Alibaba cloud intelligence brain," 2021. [Online]. Available: https://www.alibabacloud.com/solutions/intelligence-brain

[11] Clarifai, "Clarifai computer vision, nlp & machine learning platform," 2021. [Online]. Available: https://www.clarifai.com/

[12] L. Kleinrock, *Theory, Volume 2, Queueing Systems: Computer Applications*. USA: Wiley-Interscience, 1975.

[13] ——, *Theory, Volume 1, Queueing Systems*. USA: Wiley-Interscience, 1975.

[14] R. Herbrich, "Machine learning at amazon," in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM 2017, Cambridge, United Kingdom, February 6-10, 2017*, M. de Rijke, M. Shokouhi, A. Tomkins, and M. Zhang, Eds. ACM, 2017, p. 535. [Online]. Available: https://doi.org/10.1145/3018661.3022764

[15] V. Vovk, A. Gammerman, and G. Shafer, *Algorithmic learning in a random world*. Springer Science & Business Media, 2005.

[16] J. Lei, M. G'Sell, A. Rinaldo, R. J. Tibshirani, and L. Wasserman, "Distribution-free predictive inference for regression," *Journal of the American Statistical Association*, vol. 113, no. 523, pp. 1094–1111, 2018.

[17] P. Wang, Y. Li, and C. K. Reddy, "Machine learning for survival analysis: A survey," *ACM Computing Surveys (CSUR)*, vol. 51, no. 6, pp. 1–36, 2019.

[18] P. G. Ipeirotis, A. Ntoulas, J. Cho, and L. Gravano, "Modeling and managing changes in text databases," *ACM Transactions on Database Systems (TODS)*, vol. 32, no. 3, pp. 14–es, 2007.

[19] Y.-G. Jiang, S. Bhattacharya, S.-F. Chang, and M. Shah, "High-level event recognition in unconstrained videos," *International journal of multimedia information retrieval*, vol. 2, no. 2, pp. 73–101, 2013.

[20] P. Over, G. Awad, J. Fiscus, B. Antonishek, M. Michel, A. Smeaton, W. Kraaij, and G. Quenot, "Trecvid 2010 - an overview of the goals, tasks, data, evaluation mechanisms, and metrics," 2011-04-15 2011.

[21] P. Over, G. Awad, J. Fiscus, B. Antonishek, M. Michel, W. Kraaij, A. Smeaton, and G. Quénot, "Trecvid 2010 - an overview," *NIST TRECVID Workshop*, 2010.

[22] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[23] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," *Advances in neural information processing systems*, vol. 24, 2011.

[24] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization." *Journal of machine learning research*, vol. 13, no. 2, 2012.

[25] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Computers & Electrical Engineering*, vol. 40, no. 1, pp. 16–28, 2014.

[26] L. Van Der Maaten, E. Postma, J. Van den Herik *et al.*, "Dimensionality reduction: a comparative," *J Mach Learn Res*, vol. 10, no. 66-71, p. 13, 2009.

[27] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[28] V. Vovk, "Conditional validity of inductive conformal predictors," in *Asian conference on machine learning*. PMLR, 2012, pp. 475–490.

[29] G. Zeni, M. Fontana, and S. Vantini, "Conformal prediction: a unified review of theory and new challenges," *arXiv preprint arXiv:2005.07972*, 2020.

[30] H. Boström, U. Johansson, and A. Vesterberg, "Predicting with confidence from survival data," in *Conformal and Probabilistic Prediction and Applications*. PMLR, 2019, pp. 123–141.

[31] R. C. Smith, *Uncertainty quantification: theory, implementation, and applications*. Siam, 2013, vol. 12.

[32] S. Oh, A. Hoogs, A. Perera, N. Cuntoor, C.-C. Chen, J. T. Lee, S. Mukherjee, J. Aggarwal, H. Lee, L. Davis *et al.*, "A large-scale benchmark dataset for event recognition in surveillance video," in *CVPR 2011*. IEEE, 2011, pp. 3153–3160.

[33] A. Gorban, H. Idrees, Y.-G. Jiang, A. Roshan Zamir, I. Laptev, M. Shah, and R. Sukthankar, "THUMOS challenge: Action recognition with a large number of classes," http://www.thumos.info/, 2015.

[34] H. Idrees, A. R. Zamir, Y. Jiang, A. Gorban, I. Laptev, R. Sukthankar, and M. Shah, "The thumos challenge on action recognition for videos "in the wild"," *Computer Vision and Image Understanding*, vol. 155, pp. 1–23, 2017.

[35] H. Kuehne, A. Arslan, and T. Serre, "The language of actions: Recovering the syntax and semantics of goal-directed human activities," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 780–787.

[36] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, pp. 91–99, 2015.

[37] D. Greig, "Video object detection speedup using staggered sampling," in *IEEE Workshop on Applications of Computer Vision (WACV 2009), 7-8 December, 2009, Snowbird, UT, USA*. IEEE Computer Society, 2009, pp. 1–7. [Online]. Available: https://doi.org/10.1109/WACV.2009.5403129

[38] D. Kang, J. Emmons, F. Abuzaid, P. Bailis, and M. Zaharia, "Noscope: optimizing neural network queries over video at scale," *Proceedings of the VLDB Endowment*, vol. 10, no. 11, pp. 1586–1597, 2017.

[39] D. R. Cox, "Regression models and life-tables," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 34, no. 2, pp. 187–202, 1972.

[40] D. Kang, P. Bailis, and M. Zaharia, "Blazeit: Fast exploratory video queries using neural networks," *arXiv preprint arXiv:1805.01046*, 2018.

[41] N. Mehrasa, A. A. Jyothi, T. Durand, J. He, L. Sigal, and G. Mori, "A variational auto-encoder model for stochastic point processes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3165–3174.

[42] D. M. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," *arXiv preprint arXiv:2010.16061*, 2020.

[43] J. Carreira and A. Zisserman, "Quo vadis, action recognition? A new model and the kinetics dataset," in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 2017, pp. 4724–4733. [Online]. Available: https://doi.org/10.1109/CVPR.2017.502

[44] E. Vahdani and Y. Tian, "Deep learning-based action detection in untrimmed videos: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[45] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *CoRR*, vol. abs/1804.02767, 2018. [Online]. Available: http://arxiv.org/abs/1804.02767

[46] Y. Chen, X. Yu, and N. Koudas, "Evaluating temporal queries over video feeds," *Proceedings of the 2021 ACM SIGMOD International Conference on Management of Data*, 2021.

[47] ——, "Tqvs: Temporal queries over video streams in action," in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020, pp. 2737–2740.

[48] D. Chao, N. Koudas, and I. Xarchakos, "Svq++: Querying for object interactions in video streams," in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020, pp. 2769–2772.

[49] I. Xarchakos and N. Koudas, "Querying for interactions," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, 2021.

[50] D. Chao, Y. Chen, N. Koudas, and X. Yu, "Track merging for effective video query processing," in *2023 IEEE 39th International Conference on Data Engineering*, 2023.

[51] I. Xarchakos and N. Koudas, "Svq: Streaming video queries," in *Proceedings of the 2019 International Conference on Management of Data*, 2019, pp. 2013–2016.

[52] N. Koudas, R. Li, and I. Xarchakos, "Video monitoring queries," in *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 2020, pp. 1285–1296.

[53] W. N. Venables and B. D. Ripley, *Modern Applied Statistics with S*, 4th ed. New York: Springer, 2002, iSBN 0-387-95457-0. [Online]. Available: http://www.stats.ox.ac.uk/pub/MASS4

[54] C. Lee, W. Zame, J. Yoon, and M. Van Der Schaar, "Deephit: A deep learning approach to survival analysis with competing risks," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[55] Q. Ke, M. Fritz, and B. Schiele, "Future moment assessment for action query," in *IEEE Winter Conference on Applications of Computer Vision, WACV 2021, Waikoloa, HI, USA, January 3-8, 2021*. IEEE, 2021, pp. 3218–3227. [Online]. Available: https://doi.org/10.1109/WACV48630.2021.00326

[56] S. Oprea, P. Martinez-Gonzalez, A. Garcia-Garcia, J. A. Castro-Vargas, S. Orts-Escolano, J. Garcia-Rodriguez, and A. Argyros, "A review on deep learning techniques for video prediction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[57] A. Hu, F. Cotter, N. Mohan, C. Gurau, and A. Kendall, "Probabilistic future prediction for video scene understanding," in *European Conference on Computer Vision*. Springer, 2020, pp. 767–785.

[58] S. Shalev-Shwartz, N. Ben-Zrihem, A. Cohen, and A. Shashua, "Long-term planning by short-term prediction," *arXiv preprint arXiv:1602.01580*, 2016.

[59] W. Liu, W. Luo, D. Lian, and S. Gao, "Future frame prediction for anomaly detection–a new baseline," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6536–6545.

[60] Z. Liu, R. A. Yeh, X. Tang, Y. Liu, and A. Agarwala, "Video frame synthesis using deep voxel flow," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 4463–4471.

[61] S. Barocas, M. Hardt, and A. Narayanan, *Fairness and Machine Learning*. fairmlbook.org, 2019, http://www.fairmlbook.org.