C-FAR: A Compositional Framework for Anomaly Resolution in Intelligent Transportation Systems

Pavana Pradeep Kumar, Krishna Kant, *Life Fellow, IEEE*, and Amitangshu Pal

***Abstract*— In this paper, we present C-FAR, a framework for reasoning about anomalies in road-based intelligent transportation systems (ITS) based on video monitoring by the roadside camera infrastructure. The anomalies could span broad temporal and spatial ranges, including fine-grain (e.g., unsafe interactions among moving vehicles in real-time), medium-grain (e.g., aggressive/unsafe driving styles of individual vehicles over extended periods/distances), and coarse-grain (e.g., ensemble properties of the traffic over even longer time horizons). Unlike traditional approaches that utilize deep learning to recognize individual activities, C-FAR does so only for primitive movements and activities and then builds a comprehensive event logic framework. It also provides an optimal resolution of the detected/predicted anomalies by identifying the minimal changes in the controllable parameters of the system. We implemented a prototype system and tested it on three distinct real-world traffic data sets. We demonstrate that the proposed scheme can predict anomalies with over 84% recall level at 95% confidence level approximately 4.05 seconds before the incident.**

***Index Terms*— Intelligent transportation systems, anomaly detection and resolution, satisfiability modulo theories, event logic, combinatorial optimization.**

I. Introduction

With rapid advances in image processing and machine learning, it is becoming possible to monitor critical infrastructures to determine "anomalies" (or irregular behavior) in real-time to maintain safety and smooth operation. In this paper, we focus on anomaly detection in road-based Intelligent Transportation Systems (ITS) and propose a novel method based on the spatio-temporal composition of simpler activities learned through deep learning. We consider scenarios in the near future where the traffic may consist of a mix of cars with different automation levels [1] and fuels, including the rapidly advancing Electric Vehicle (EV) technology.

A. Current Work on Anomaly Detection and Its Limitations

Anomaly detection is a well-researched subject but primarily focused on detecting specific situations. Several studies attempt to detect anomalies automatically as "outlier" events, e.g., walking/riding the wrong way on a one-way path [2]. The current trend in the field is to capture even more complex activities with more sophisticated deep-learning (DL) models, as described in the related work section (§II). The most widely used techniques utilize Recurrent Neural Networks (RNNs) – more precisely, Long Short-Term Memory (LSTMs) – either alone or in conjunction with Convolutional Neural Networks (CNN) to learn from data sequences and to capture spatial information about road networks in addition to capturing long-term temporal patterns. The high complexity of these models results in three key drawbacks: (a) Models such as LSTMs require an enormous amount of data to train, and such data is hard to come back, especially in the context of anomalies, (b) A separate model needs to be constructed and trained for each class of activities, which makes the method unscalable to complex environments, and (c) the accuracy of most of these models is rather poor, with standard error in ten's of percent, which is unacceptable for real-world deployment of such an approach.

Manuscript received 10 October 2021; revised 31 March 2022; accepted 10 July 2022. Date of publication 22 August 2022; date of current version 26 January 2023. The Associate Editor for this article was M. Dabbaghjamanesh. *(Corresponding author: Pavana Pradeep Kumar.)*

Pavana Pradeep Kumar and Krishna Kant are with the Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122 USA (e-mail: pavana.pradeep@temple.edu; kkant@temple.edu).

Amitangshu Pal is with the Computer Science and Engineering, Indian Institute of Technology Kanpur, Kanpur 208016, India (e-mail: amitangshu@cse.iitk.ac.in).

Digital Object Identifier 10.1109/TITS.2022.3196548

B. Our Contributions

In this paper, we take a completely different logic-based approach. Our proposal, C-FAR, does not try to detect anomaly directly (e.g., an accident occurring or about to occur); instead, it learns about simpler events relevant to the anomaly (e.g., two distances between two vehicles becoming less than a threshold). We then combine these based on domain knowledge (e.g., motion physics) and spatio-temporal reasoning to determine the presence of an anomaly. The logic formulation provides considerable flexibility and can efficiently encode the physics of the situation, which in turn can be used to predict the anomaly occurrence sufficiently in the near future to enable corrective action. We use the same formulation to frame an optimization problem to resolve the anomaly with minimally acceptable perturbations in the rules. The resulting "correctional advice" can then be provided to the driver/driving system for the application. This framework still needs deep learning to drive object recognition and proximity. Still, this technology is well-established, and the tasks can be done quickly and with a high degree of accuracy.

Compared with existing RNN-based methods [3], [4], the C-FAR framework offers: (a) Reduction of the problem to learn a core set of primitive events/activities through deep learning, which can be fast and accurate, (b) a highly flexible way of combining primitive events/activities, along with the relevant physics, to detect and resolve anomalies, and (c) extensibility as the system or anomaly definitions evolve. We evaluate our approach on several datasets and show that it can detect and predict an anomaly with an accuracy of up to 91% and a recall rate of 85%. Additionally, it can anticipate anomalies much earlier (4.05secs vs. 2.15secs [3]) and resolve 100% of anomalous cases.

To the best of our knowledge, this is the first paper that attempts to predict complex events from a composition of simpler events where the weights are not fixed in advance.

C. Paper Organization

The remainder of this paper is organized as follows. Section II discusses the related work. Section III discusses anomalies in ITS. Section IV discusses formal modeling of the anomalies. Section V presents the proposed framework. Experimental evaluation and results are summarized in section VI. Section VII then concludes the paper.

II. Related Work

In [5] authors have proposed a single-class neural network technique composed of Convolutional Auto Encoder (CAE) to extract robust spatiotemporal features for detecting abnormal events in crowded scenes. Several deep learning methods have recently surfaced as non-parametric alternatives for anomaly prediction. Major literature survey papers on DL techniques in traffic flow analysis/prediction are contained in [6] and [7]. Another survey paper [8] discusses deep learning methods for anomaly detection in surveillance videos in detail, including open problems and analysis of supervised and unsupervised methods. There are several works that distinguish abnormal actions and commands issued over a distributed network of unmanned Autonomous Vehicles (AVs). Authors in [9] propose an autonomous intrusion detection scheme for discovering advanced and sophisticated cyberattacks that exploit drone networks using Machine Learning (ML) algorithms like decision trees, k-nearest neighbors, naive Bayes, support vector machines, and deep learning multi-layer perceptrons. In [10] authors present a multi-stage intrusion detection framework to identify intrusions from ITS. The proposed framework is based on normal state-based and bidirectional Long Short Term Memory (LSTM) architecture to efficiently discover intrusions from the fundamental network gateways and communication networks of AVs. However, the activity complexity and data limitations generally yield accuracies that are not good enough for real-world use. Also, much of the research focuses on a small geographic area or a brief period, raising concerns about scalability [11]. Another major challenge in deploying a hybrid deep learning model that incorporates both spatial and temporal modeling is the distinction between training and the time horizons used for prediction. While some research has demonstrated that hybrid deep learning architectures can enhance performance in certain conditions [12], others continue to debate the necessity and efficacy of fine-tuning such models.

III. Anomalies in Road-Based Its

A. Case for Video-Based Anomaly Detection

We assume the deployment of Road-Side Infrastructure (RSI) to comprehensively monitor the road segments through smart cameras mounted on every light pole. Each camera monitors the traffic in its view and does some simple image processing tasks such as object detection and tracking in each frame and data transmission to the next level, often known as the *Road-Side Edge Controllers (RECs)*. The tracking algorithms can also avoid transmission of redundant frames and further adapt to the available transmission bandwidth as in our earlier work [13]. The RECs receive video streams from multiple cameras along a road segment and use them for flexible monitoring of activities and anomalies.

Such a system can augment the increasing array of safety features in the vehicles, commonly known as ADAS (Automated Driver Assistance Systems). Each vehicular ADAS considers safety from a local perspective; instead, the monitoring RSI can provide a global perspective and warn the vehicular system or the driver accordingly. A similar mechanism can also be used in other environments such as hospitals, senior care centers, factories with only a few workers, etc.

Given the increasing processing power in smart cameras, the detection/tracking of essential objects can be done by camera itself. Further processing, including perspective transformation and estimation of orientations and speeds of the objects may be done by the camera itself or by the RECs. The REC can then build a spatio-temporal logic model of the situation that includes all "facts" of the anomaly situations and the supporting "theories" (i.e., Newton's laws, arithmetic, etc.).

B. Type of Anomalies

In general, anomalies can occur at multiple temporal and spatial scales. In ITS, we can identify fine-grain, medium-grain, and coarse-grain anomalies. A *fine-grain* anomaly refers to the interaction between adjacent vehicles or vehicles and other objects (e.g., pedestrians, objects on the road, etc.) with safety implications. This includes the relative movements (e.g., distance and how it changes) that may damage, injure, or loss of control. They are the most critical and the most difficult to handle since they must be predicted quickly and yet provide at least 1-2 seconds for corrective action to be taken.

The *medium-grain* anomalies concern individual vehicle's behaviors that are detrimental to the safety and efficient driving (e.g., weaving through the traffic, persistent acceleration/deacceleration pattern, etc.). Note that monitoring such anomalies requires behavior over longer periods (e.g., minutes) and spatial spans (e.g., 100s of meters). Automated vehicles that have been compromised due to security attacks or HW/SW malfunctions could show an even broader range of abnormal behavior, thereby making automatic detection crucial. Finally, the *coarse-grain* anomalies, which concern congestion on the road, feeder roads, and at fueling stations. These anomalies could have extended relevance, such as the congestion at multiple EV charging stations in an area could imply stress the smart grid.

We largely focus on fine-grain and medium-grain anomalies in this paper. The fine-grain anomalies include: *rear-end or front-end collisions*, *intersection collisions*, *lane-change collisions*, and collision with pedestrian/animal on the road. The medium-grain anomaly pertains to *aggressive driving behavior* including speeding, weaving in and out of traffic, running through pedestrian crosswalks, etc. For fine-grain monitoring, the interest is primarily in detection conditions that may cause a collision rather than the collision itself.

C. Challenges in Compositional Anomaly Detection

The anomaly detection problem can be formulated as a Boolean satisfiability problem so that the popular SMT (satisfiability modulo theory) based tools can be used along with suitable theories as stated above. However, most of the assertions involved depend on time and space, which can be expressed using a temporal logic such as Linear Time Logic (LTL). However, a plain LTL expression makes a definitive statement about the future (e.g., an assertion holds until some event happens). Instead, we need a new notion of a *fluent* whose validity can change at any time instant. There are several temporal logics built using fluents and assertions [14], [15]. A popular framework for this is *Event Calculus* [16] that we use here. Another calculus is Situation Calculus (SC) [17], but it specifies a sequential occurrence of actions and does not allow time evolution.

The order in which traffic-related events occur and the relationships between successive traffic situations may have an effect on whether an anomaly occurs. To capture this, the model should represent event preconditions, time-dependence of events, including absolute event times, and their impact on the traffic flow. Event Calculus (EC) provides constructs to reason about situations, events, and changes in time, allowing a precise specification of time relationships between situations and events, which is critical for describing traffic anomalies. In addition, concurrent actions can also be specified in EC.

Since numerous factors influence the traffic events and hence anomalies in the real world (e.g., weather, road conditions, traffic controls/regulations, driver behavior, etc.), the assertions in the EC formulation would generally be based on these aspects, even though direct modeling of these factors is not needed. For example, speeds can be assumed to be upper bounded and consistent with the macroscopic fundamental diagram [18].

IV. Formal Activity Modeling

A. Defining Events and Fluents

The main entities defined in the EC are as follows:

- *Events:* actions that occur at a certain point in time.
- *Fluents:* entities that modify their state as a result of occurrence of event or an action. Each fluent can be initiated or terminated by multiple events and has a time duration.
- *Predicates:* entities that specify when events occur or the state of the fluents at various points in time. The EC specifications include a basic set of predicates. However, extra predicates can also be defined based on the requirements of the user.
- *Constraints:* take the form of rules that define the relationship between fluents and events.
- *Domain independent axioms:* are the default formalism logics that define the relationship between the predicates.

TABLE I

Table: Main Predicates of RTEC

Predicate	Meaning
hA(E, T)	Event E *happens at* time T
hF(E, I)	Even E *happens for* I intervals
in(F=V)	*Initial* value of fluent F =V at time 0
hoA(F=V, T)	Value of fluent F *holds at* V at time T
hoF(F=V, I)	Value V of fluent F *holds for* I intervals continuously
inA(F=V, T)	Fluent F with value V *initiated at* time T
tA(F=V, T)	Fluent F with value V *terminated at* time T

In our work, we use an efficient dialect of the Event Calculus, termed "Event Calculus for run-time Reasoning" (RTEC) [19]. RTEC is an open-source implementation of the EC in Prolog and uses LTL with integer time points. RTEC implements novel techniques for identifying complex events from a set of primitive events that are also scalable to large volumes of complex events. Simple implementations of EC are time and memory-intensive, making them unsuitable for developing real applications. This is because each time the EC engine is queried, the computation is restarted, and the validity intervals for all *fluents* are recalculated. RTEC addresses these issues by developing a technique for caching the results of sub-computations. Additionally, the indexing technique used in RTEC makes it disregard data streams that are irrelevant to the current queries.

An event description in RTEC contains rules that define the event instances using the *hA* and *hF* predicate. The *fluents* that are time-varying properties and the effects of events on *fluents* are defined using the *inA* and *tA* predicates. The value of *fluents* at any time point is defined using the *hoA* and *hoF* predicates. If F is a variable ranging over *fluents*, the term $F{=}V$ denotes that variable F has a value V. There also exists Boolean fluents with values *true* or *false*. The Table I shows the predicates used in RTEC tool.

B. Detecting and Predicting Anomalies

Identifying anomalies through a spatio-temporal assertion checking can be regarded as "detection" when a specific event such as a collision is imminent or significant traffic congestion has already occurred. However, if we can anticipate the event well enough in advance, we can do it as a "prediction". There is an apparent tradeoff between the accuracy of prediction and how far in advance the prediction can be made. Nevertheless, since some level of false positives is preferable to the anomaly actually occurring, an earlier prediction is usually preferable. The prediction is necessarily limited to the factors that are comprehended by the model; for example, without an accurate model of driver actions, we cannot predict what the driver might do in a certain anomalous situation. However, to the extent that such knowledge is available, it can be easily included in our framework.

In reality, detecting an anomaly cannot always be reduced to a simple Boolean condition to be satisfied. Instead, the anomaly is usually characterized as a confluence of several conditions of varying importance. As a trivial example, if d_x is the distance to the next vehicle in the same lane, and d_y is the clearance between vehicles in adjacent lane, we may want to associate higher importance to $d_x < x_0$ (i.e., d_x going below the critical value x_0) than to $d_y < y_0$ i.e., d_y going below the critical value y_0). One way to achieve this is to associate a *weight*, say w_i, with the ith condition, and use $\sum_i w_i > W_0$ as an indication of anomaly, where W_0 is a predefined threshold.

In general, some conditions (the most important ones) may be considered as *hard* in that they must hold individually. In contrast, others (the less important ones) may be regarded as *soft* in that they are considered together through their overall weight. It is also likely that the weights are not static but depend on various spatio-temporal aspects and context (e.g., day vs. night time, roads with different speed limits, etc.) This can be handled except that a weight change would require that we pause all current condition evaluations, change all weights that need to be changed simultaneously, and then resume the evaluation.

To handle hard/soft conditions, we can use an extension to the Boolean Satisfiability problem known as *Weighted Partial Maxsat* (WPM2). In standard WPM, we have a Boolean formula expressed in the Conjunctive Normal Form (CNF), where each clause is expressed in terms of some Boolean variables. Let $Y = \{y_i, i = 1, 2, \ldots\}$ denote the set of Boolean variables in the formula. Then each CNF clause will include a subset of these in normal or negated form. In WPM, each clause is designated as either *hard* or *soft* with a given weight. We then have an optimization problem to find an assignment that satisfies all hard clauses and minimizes the total weight of soft clauses. Solution methods for such problems are currently a hot topic, with many approaches, and are a prominent part of the annual MaxSAT competition [20]. The solvers include both "complete" methods (e.g., deterministic search) and "incomplete" ones (e.g., combinatorial optimization based), and we use the latter approach as it caters to detection and resolution.

In our problem, the Boolean variables y_i's represented conditions involving some underlying state variables of the system. For example, $y_1 = (speed < 50)$ where $speed$ is the underlying real-valued state variable. Standard WPM has no notion of such underlying variables and thus does not consider them in the value assignment. In our solution, we will perturb these *Underlying State Variables* (USV's), henceforth denoted as v_j, $j = 1..J$ to distinguish them from the *Boolean variables* (BV's) y_i's.

C. Resolving Anomalies

When applied to the optimization problem as in IV-B, a WPM2 solver would return an UNSAT core C' in case of anomalies. The UNSAT core consists of the rules and the clauses involved in the anomaly. For resolving the anomaly, we need to perturb the underlying variables of the unsatisfiable clauses so that they become satisfiable. For example, a clause such as (speed $< 60 \vee$ distance > 2) will be treated by WPM2 as the CNF clause $(x_1 \vee x_2)$ where $x_1 =$ speed < 60 and $x_2 =$ distance > 2, and (x_1, x_2) denotes the clauses and associated variables that are perturbed to resolve an anomaly. *To the best of our knowledge, all work on WPM2 stays strictly at the Boolean level and has no knowledge of the underlying conditions or variables.* We address this through a combinatorial optimization approach, for which numerous algorithms exist and we found that Dynamically Dimensioned Search (DDS) [21] generally works the best. It initially perturbs most variables but fewer of them as the iterations proceed, akin to temperature decrease in Simulated Annealing.

The key to finding a near-optimal solution quickly in combinatorial optimization algorithms is the use of *situations or ongoing events*. The perturbations to the solution made at each stage represent various correlations and dependencies. Hence to resolve an anomaly, the situation-based DDS intelligently perturbs the variables, i.e., based on current situations or events, and selects the clause to perturb that has the most significant influence on the anomaly. Note that only the vehicular parameters (e.g., speed, acceleration, orientation, etc.) are perturbable parameters for anomaly resolution; we cannot change anything about other objects that the vehicles interact with (e.g., pedestrians, animals, static objects, etc.) cannot be perturbed.

One other unique aspect of anomaly resolution is that it is a constrained optimization problem, with the constraint being the satisfiability of the modified formula that ensures the anomaly no longer exists. In this paper, we use the epsilon-constraint satisfaction method, which appears to work quite well in practice. This method essentially quantifies the notion of "degree of feasibility" based on how close the proposed solution is to the constraint boundaries [22]. This is then used along with the current cost to decide how to perturb the variables.

V. Compositional Framework for Anomaly Resolution

Fig.1 shows our overall framework, named C-FAR, which consists of two stages. The first stage is a lightweight object detection model using a convolutional neural network (CNN) model called YLLO that we have developed for video analysis in [13]. YLLO stands for You Look Less than Once and refers to the highly efficient processing of video sequences. The second stage for each detected object (e.g., vehicles, pedestrians, etc.) is a spatio-temporal logic-based reasoning system that captures the relative movements of the objects in real-time to detect/resolve anomalies using a combinatorial optimization-based approach.

A. CNN Based Object Detection and Tracking

YLLO is a lightweight object detection technique based on YOLOv4 and is optimized for continuous video streams by utilizing redundancy to identify the "only" essential frames. YLLO is a three-stage process that begins with a scene change detection algorithm and progresses to object detection via YOLOv4 or any single shot detector. The Simple Online and Real-time Tracking (SORT) algorithm assign a tracker to each detected object or multiple objects. YLLO decouples classification and regression tasks to eliminate redundant objects between the frames. Additionally, before sending frames to object detection, for the scene change detection, it generates Color Difference Histograms (CDH) for edge orientations, where edge orientations are determined using the Laplacian-Gaussian edge detection framework.

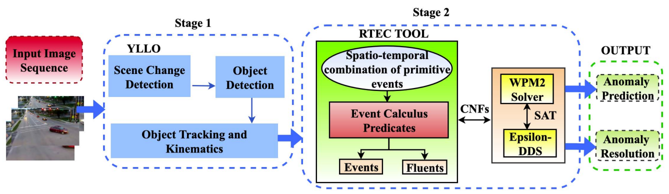

Fig. 1. Compositional framework for anomaly resolution (C-FAR).

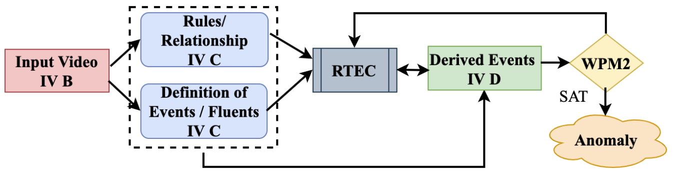

Fig. 2. Stage 2 flowchart.

B. Input to RTEC Tool

In stage 2 of the C-FAR framework, the RTEC tool receives input as EC predicates representing time-stamped primitive activities detected on individual video frames as shown in Fig. 2. For example, the object's bounding box coordinates can define the appearance of a static object or multiple moving objects in each frame. Additionally, we have the angle/orientation of the object and the direction in which they are moving.

The primitive events are defined along with their associated timestamps, which indicate the time-point in which the activity occurred. The *hA* predicate establishes this type of input. For instance, *hA(emergencyBreaking(id6, 60)* indicates that an object(id6) engaged in emergency braking at video frame 60. These primitive activities are represented as events in the EC and we use the *inA* and *tA* predicates for expressing the conditions in which these events initiate and terminate an anomaly described above.

The tracked people or object's coordinates are specified as 'X' and 'Y' pixel positions at each time point. These coordinates are expressed using the *hoA* predicate, for example, *hoA(coord(id2) = (14, 55), 40)* indicates that the coordinates of object with id2 are (14, 55) at time point (frame number) 40. The *hA* predicate represents the first and last time points a person or object is detected, given by "detect/exits". For example, *hA(detect(id2), 17)* indicates that an object with the id2 is detected for the first time at time-point (frame number) 17. Similarly, if an object disappears from a frame, *hA(exits(id2), 560)* indicates that the object2 leaves the scene at time-point (frame number) 560.

C. Formation of Events and Fluents

Anomalies or complex events are represented using EC fluents defined mostly with hoA predicate, which can also compute the associated intervals. For example, *hoA(following(id1, id3) = true, [(0, 40),(340, 380)]* indicates that object1 was following object3 during the intervals (0, 40) and (340, 380).

A few examples of events defined include, (1) *becomesSafe (v1, v2)* that denotes the separation or distance between vehicle *v1* approaching or following vehicle *v2* becomes safe, similarly, (2) *becomesUnsafe (v1, v2)* that is opposite of the event becomesSafe, (3) *follow (v1, v2)* that denote vehicle *v1* starts following vehicle *v2*. Some examples of fluents defined include, (1) *reducedSpeed (v)*: vehicle *v* reduces its speed, (2) *laneShift (v)*: vehicle *v* is changing a lane as a result of steering, (3) *collision (v1, v2)*: vehicle *v1* has collided with vehicle *v2*.

After defining events and fluents, we must define an initiation and termination map for each defined fluent in the system, indicating which events initiate and terminate which fluents. The next step is to specify the relation between fluents and events in the form of rules. For example, the initiation and termination map for fluent *unsafeSeparation* is shown in Definition. 1.

In the ruleset, *unsafeSeparation*, *speed* and *direction*, are the input events and *th* is an temporal predicate indicating numerical threshold of traffic patterns and in this case it represents the user-specified distance and speed threshold. The rule set defined states that *unsafeSeparation(v1)* is a Boolean fluent, which is invoked when a *speed* event is detected for a vehicle v1 and v2 where v1 is following v2 in the same direction and the distance between two vehicles (d_{close}) is less than a predefined threshold D_θ, and v1's speed is greater than v2's speed. The event *unsafeSeparation(v1)* is terminated when the vehicle v1 is not following v2 and v1's speed is lesser than v2's speed and the distance between two vehicles (d_{close}) exceeds D_θ, or when either the vehicle v1 or vehicle v2 exists from the scene reported by *exits(v1)* or *exits(v1)* event. Similarly, we have defined the initiation and termination map of other fluents used in the system.

D. Derived Events

When dynamic dependencies are significant, such as when modeling accidents in which traffic conditions interact dynamically (e.g., a braking maneuver is executed in response to danger), developing a system capable of representing dynamic relationships is necessary. We define the derived events, i.e., the events that occur due to the change in state or value of another fluent and/or the occurrence of another event. These events indicate when specific actions occur in traffic due to a combination of particular conditions. They assist us in modeling the dynamic behavior of accident scenarios and the effects of introducing new actions.

At each time slot, before invoking the WPM2 solver, we find relevant rules corresponding to derived events based on current ongoing events [23]. We identify the rules or relations **R'** $\subseteq$ **R** that lead to derived events based on the current events, either directly or indirectly. The dependency is expressed via a dependency graph G, where the vertices denote the rules/relations, and the (directed) edges denote the dependency between them. We then take the transitive closure of G (say G') using the Floyd–Warshall algorithm. Thus, in G' an edge $i \rightarrow j$ denotes that j is directly or indirectly dependent on i. The rules **R'** expressed in CNF are then passed on to a WPM2 solver [24].

E. Anomaly Detection and Prediction

inA(unsafeSeparation(v1, v2, d_{close}*),t* $\leftarrow$ *hA(distance(v1, v2,* d_{close}*),t)* $\wedge$ *th(*$d_{close} < D_\theta$*)* $\wedge$ *hA(speed(v1,* S_{v1}),t) $\wedge$ *hA(speed(v2,* S_{v2}),t).

tA(unsafeSeparation(v1, v2, d_{close}),t $\leftarrow$ *hA(distance(v1, v2, d_{close}),t)* $\wedge$ *th($d_{close} \geq D_\theta$)* $\wedge$ *hA(speed(v1, S_{v1}),t)* $\wedge$ *hA(speed(v2, S_{v2}),t).*

tA((unsafeSeparation(v1, v2, d_{close}),t) $\leftarrow$ *hA(exits(v1),T)* $\vee$ *hA(exits(v2),T).*

Definition 1: Initiation and Termination of fluent *unsafeSeparation*

(a) Relationship between events and fluents in a rear-end accident scenario

inA(becomesUnsafe(v1, v2), t) $\leftarrow$ *hoA(speed(v1, S_{v1})), t)* $\wedge$ *hoA(speed(v2, S_{v2})), t)* $\wedge$ *hoA(unsafeSeparation(v1, v2, d_{close}), t)* $\wedge$ *hoA(acceleration(v2, A_{v2}, t)).*

(b) Definition of Fluent Collision

hoA(collision(v1, v2), t) $\leftarrow$ *hoA(following(v1, v2), t)* $\wedge$ *hoA(becomesUnsafe(v1, v2), t).*

Definition 2: RTEC Rules Set

Given the inputs and the derived events, our system should be able to detect and predict the anomalies as mentioned in IV-B. For example, to predict if a collision can happen in the next few slots, we can determine the distance between two tracked objects and compare the distance with predefined thresholds. For example, *hoA(safeDistance(id1, id3, 30) =true, 80)* states that object1 is "close" to object3 at time 80 and also their distance is at most 30 pixel positions. Further, we can also compute the validity of fluents given by the maximal intervals for which two tracked objects are "close". For example, *hoF(safeDistance(id1, id5) = true, [(20, 30)]* states that (20, 30) is the maximal interval for which the distance between object1 and object5 is safe. Similarly, we can find the maximum validity interval for the unsafe distances between two vehicles. In general, it is not just the distance that matters but also its increase or decrease in the traveling rate.

Let us consider an example that represents temporal constraint relationship between the event *becomesUnsafe(v1, v2)* and the fluents *distance(v1, v2)*, *speed (v, S_v)*, *acceleration(v, A_v))* and few other fluents that identify a rear-end accident scenario that can be defined as shown in Definition. 2(a).

The event *becomesUnsafe(v1, v2)* is a long term activity expressed as a Boolean event in terms of other primitive activities defined using *hA* predicate and based on the spatial information given by coordinates and orientation of the tracked objects in the traffic scene. *becomesUnsafe* is triggered when two vehicles are following each other from behind, traveling in the same direction, and the distance between them, d_{close} is less than a predefined threshold triggered by event *unsafeSeparation*. The ruleset specifies a predefined speed limit using *th* against which the vehicle's current speed is compared. The calculation of distance and acceleration values are based on classical kinematic equations given in [25]. Further, we can determine the fluent collision holds in any given traffic scene, which is defined as shown in Definition. 2(b).

F. Anomaly Resolution

Following the prediction of an anomaly in stage 2 of the C-FAR framework, the ϵ-DDS algorithm is supplied with clauses used to determine the anomaly. Let v_j, $j = 1..J$ denote the J USVs that appear in the UNSAT core C'. Let $\mathbb{N}$ denote the set of clauses in the C'. Also, let $\eta(C'_i)$ denote the set of variables in the clause C'_i. We assume that if variable v_j originally has the value τ_j, its modified value is denoted as $\hat{\tau}_j$.

Let w_i denote the weight of the unsatisfiable clause C'_is. The notations $v_j^{(min)}$ and $v_j^{(max)}$ indicate the minimum and maximum values of the variable v_j. We assume that $\mathbb{H}$ denotes the set of hard clauses in the UNSAT core (obviously a subset of $\mathbb{N}$). The goal is then to determine the modified τ_j's, denoted here as $\hat{\tau}_j$, by solving the following optimization problem:

$$\text{Min} \quad \sum_{i\in\mathbb{N}} \forall_{j\in\eta(C'_i)}\ g(\hat{\tau}_j - \tau_j) \times w_i \tag{1}$$

$$\text{s.t.} \quad \forall_{i\in\mathbb{H}}[C'_i(\forall_{j\in\eta(C'_i)}\hat{\tau}_j) = \text{true}]] \tag{2}$$

$$\forall j, \quad v_j^{(min)} \leq v_j \leq v_j^{(max)} \tag{3}$$

$$|\mathbb{H}| \leq \sum_{j=1}^{J} \mathbb{I}(\hat{\tau}_j - \tau_j) \leq |\mathbb{H}| + L \tag{4}$$

The objective function here is the total weighted perturbation, with the function $g()$ indicating the suitable measure of the magnitude of the perturbation. Since both positive and negative differences should be counted, the simplest choice is $g(x) = |x|$. A more common choice, as used in [23] is $g(x) = x^2$. The latter, however, discourages larger perturbations and thus would tend to perturb several variable by small amounts (instead of a few variables by larger amount). This may be fine for an automated driving system but less desirable for human drivers who would do better at managing only 1 or 2 variables.

The first constraint says that all the hard clauses become true following the perturbation. It is assumed that for each variable j that is not perturbed, $\hat{\tau}_j = \tau_j$. In the last constraint, L is a small number (say, 1 or 2, in most cases) and represents the number of soft clauses that can be perturbed, and $|\mathbb{H}|$ is the number of distinct underlying variables in the hard clauses. The last equation ensures that we perturb all variables required to satisfy hard clauses, but very few others. The function $\mathbb{I}$ is the index function (defined as 0 if the argument is zero, else 1).

Situation based ϵ-DDS: Consider an objective function $f(x)$ of input vector x, along with a set of constraints $\phi_i(x), i = 1, 2, \ldots, K$. Let $\sigma_i(x) \in [0, .., C_i]$ denote a cost measure for constraint $\phi_i(x)$ that indicates to what extent the constraint is violated for input vector x. By definition, if the constraint is satisfied, then $\sigma_i(x) = 0$. Let $\sigma(x) = \sum_{i=1}^{K} \sigma_i(x) / \sum_{i=1}^{K} C_i$ denote the overall normalized cost of violating the constraints, which has the range $[0, .., 1]$. Now consider an existing solution x_1, and new proposed solution x_2. The ϵ comparison between them defines a specific way of determining if x_2 is better than x_1 by considering both the objective function and the constraints. In particular, suppose that the objective is to minimize the objective function. Then the "epsilon less than" relationship between x_2 and x_1, denoted $x_2 <_\epsilon x_1$ is defined

TABLE II
CHARACTERISTICS OF DATASETS USED

Dataset	Total No of Videos	No of frames /Video	Videos with variations in Camera Resolu.	View Depth	Weather Cond.	Activity FG/MG
DAD	1730	100	✗	✗	✗	✓/✗
CADP	1416	366	✓	✓	✓	✓/✗
TU-DAT	280	960	✓	✓	✓	✓/✓
AI-City	250	2400	✓	✓	✗	✓/✗

as follows: [22]:

$$x_2 <_\epsilon x_1 \Leftrightarrow \begin{cases} f(x_2) < f(x_1), & \text{if } \phi(x_2),\ \phi(x_1) < \epsilon \\ f(x_2) < f(x_1), & \text{if } \phi(x_2) = \phi(x_1) \\ \phi(x_2) < \phi(x_1), & \text{otherwise.} \end{cases} \quad (5)$$

The intuition behind this comparison is that, if the solutions x_1 and x_2 are feasible, mostly feasible (as determined by ϵ), or having the same sum of constraint violations (the number of unsatisfied constraints in our case), then they are compared using their objective values $f(x_1)$ and $f(x_2)$. Otherwise, if both x_1 and x_2 are infeasible, they are compared based on their sum of constraint violations.

VI. EXPERIMENTAL EVALUATION

In this section, we evaluate our C-FAR framework on four real-world datasets, including our collected TU-DAT dataset and three public datasets. i.e., Dashcam Accident Dataset (DAD) [3], Car Accident Detection and Prediction (CADP) dataset [26] and NVIDIA AI City Challenge 2021 [27]. The characteristics of the datasets are shown in Table II. Columns 2-3 in the table list the size of the dataset in terms of number of videos and number of frames/video. The last four columns indicate if the dataset has the following features: (a) images captured with cameras of varying resolutions, (b) varying distance between camera and the vehicle of interest, (c) different weather conditions (foggy, sunny, snowy, day/night, etc.), and (d) whether fine-grain (FG) and medium-grain (MG) activities are represented (see section III-B). The model is validated through a comparison of state-of-the-art methods. The experiments were performed on a computer with Intel(R) Core(TM) i7-7700 CPU @ 3.60 GHz, 32 GB RAM, and 1 TB SSD and SWI-Prolog 8.2.3.

A. TU-DAT Data Collection

TU-DAT Dataset: In this paper, we collect a challenging dataset titled Temple University - Data on Anomalous Traffic (TU-DAT) in order to improve the accuracy of accident detection in ITS. TU-DAT, in particular, contains a diverse set of accident types, weather conditions, and videos collected in challenging environments, enhancing the self-adaptability of accident detection methods in a variety of traffic situations.

We developed a crawler written in Python to scrape the accident videos from news reporting and documentary websites. We also searched YouTube videos for each type of anomaly using text search queries (with slight variations, e.g., “unexpected object on the road”, “pedestrian accident,” etc.). To ensure that our method applies to roadside edge devices, we use only footage and images from traffic CCTV cameras.

TABLE III
STATISTICS OF TU-DAT DATASET

Conditions	**#frames**	**Accident Types**	**#frames**
Day light	9796	Weaving thru traffic	2417
Night/low light	1487	X-section accidents	6566
Foggy	445	Tailgating/Driving Maneuvers	1452/305
Rainy/Snowy	128/274	Highway/Rear-end Accidents	1254/1215
Camera too far	211	Pedestrian accidents	447

We have collected around 210 videos varying around 24-30 FPS of road accidents through these steps with 17255 accident keyframes and 505245 regular frames. The details of our dataset is shown in Table III.

Additionally, as an alternative to real-world videos, various AI-powered game simulations feature realistic graphics and intelligent car bots that may aid traffic analysis and accident detection. The authors in [28] proposed a 3D CNN-based model for detecting accidents, which they validated using YouTube and Grand Theft Auto (GTA) videos. Their model performs 10% better when trained on GTA game videos than when trained on YouTube videos. Given the difficulty of obtaining real-world traffic videos to analyze aggressive driving, we adapted the BeamNG.drive [29] game simulator to generate road traffic video data to simulate aggressive driving behaviors such as speeding, tailgating, weaving in and out of traffic, and running red lights. We gathered approximately 40 videos of positive examples and 25 videos of negative examples.

We utilize Computer Vision Annotation Tool (CVAT) [30] to annotate the video frames. For temporal annotations, the anomalous situation time is labeled at the time when an anomaly happens. The TU-DAT dataset is made available for research use and can be found in GitHub.[1] Fig. 3(a)-(b) illustrate a crash scenario at an intersection, where the car and van in (a) have a safe separation, but the car is traveling as fast as the van, resulting in the crash situation depicted in (b). Similarly, Fig. 3(c)-(d) illustrate a collision in which a vehicle collides with an electric pole on the road as a result of the vehicle’s abrupt steering.

B. Evaluation Metric

We evaluate our proposed model based on the correctness of anticipating a future accident or anomaly. Given an input, our method calculates the confidence of anomaly at every time slot using metrics like object distance or orientation which are defined in terms of fluents or events. When the confidence is greater than or equal to a threshold δ at time-slot t, the C-FAR framework affirms that an anomaly will occur in the future. If the input video contains the footage of an accident, this is a True Positive (TP) anticipatory signal. Hence the accident is correctly anticipated at time-slot t, which is $t' - t$ time-slots before it occurs at t'. This could be a False Positive (FP) anticipation if the input video is not an accident video. If, on the other hand, the confidence levels across multiple time-slots are less than the δ threshold, the method asserts that no accident will occur in the future. If the input video is of an accident, it is a False Negative (FN) prediction; otherwise, it is a True Negative (TN) prediction.

[1]https://github.com/pavana27/TU-DAT

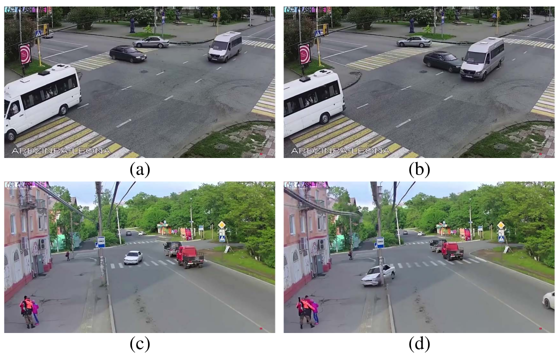
Fig. 3. (a)-(d) Some frames of TU-DAT dataset of accident scenarios.

Given these, we can compute the estimates of 3 key performance metrics: accuracy, precision, and recall which are calculated using standard metrics.

Of these three metrics, the most important one is Recall since it relates to the fraction of actual accidents (anomalies) predicted by our mechanism as well. The precision is not quite as crucial since it is okay to have some false alerts. Nevertheless, the purpose of the proposed mechanisms is to improve precision over what is possible from the ADAS capabilities in individual vehicles.

C. Results and Discussion

Fig. 4 illustrates how our C-FAR framework can be used to evaluate a variety of accident scenarios, including rear-end collisions, intersection collisions, and a car colliding with a pedestrian. The scenario involves six vehicles and a pedestrian. The sequence of events is as follows. At time t0, the vehicles v3, v4, and v5 follow each other at a safe distance, and at t1, v5 suffers a breakdown, executes a pullover, and begins emergency braking. After a few time slots, vehicle v4 changes the lane, v3, unaware of this sudden driving maneuver executed by v4, collides with vehicle v5. Similarly, v1 and v2 are at a safe distance and traveling perpendicular to each other when they collide after a few time points. Furthermore, v6 traveling faster than the posted speed limit hits a pedestrian P1 crossing the street. We have represented collision as a fluent that holds from the time it occurs, and thus we can query this fluent at any time moment. The following Prolog syntax illustrates an example query about a collision between vehicles in a traffic scene that is evaluated to be true:

?- hoA(collision(v3,v5),12). **yes**

In addition to that, we can also determine if certain events occur and whether certain fluents persist at specific time points. For example, we can perform query computations to determine the value of the fluents at any point in time, such as the vehicle's velocity, and calculate the distance between any vehicles while deciding whether they are in a safe separation from one another or not. Additionally, we can also determine whether or not an event occurred based on the validity of the fluent initiated by the event; for example, to determine whether or not an anomaly such as a rear or pedestrian accident is occurring. Some examples of queries that can be executed are given below:

?- hoA(distance(v3,v4,D2). **D2=21? yes**
?- hoA(intersectCollision(v1,v2),22). **yes**
?- hoA(becomesUnsafe(v3,v5,78). **yes**
?- hoA(safeDistance(v3,v5,75). **no**
?- hoA(pedestrianAccident(p1,v6),42). **yes**

The above example of queries shows the analyses of the effect of various parameters on the evolution of the traffic scenario, such as estimating the distance between vehicles v3 and v4 at any point in time shown in Fig. 4. Analyses of certain parameters are essential in determining when and how to act to avoid accidents. We could also determine the moment when vehicle v5 has stopped. Additionally, we demonstrated various analyses performed on the proposed system, including estimating safe and unsafe separation between vehicles v3 and v5, estimating an impending collision between vehicles v1 and v2 at the intersection, and estimating a collision with the pedestrian. Similarly, we can determine whether a car is following another, the vehicle's velocity at any point in time, etc.

1) Accuracy and Precision-Recall Results: Fig. 5 shows the Accuracy and Precision-Recall (PR) values of the proposed system evaluated on the DAD dataset and on our own dataset TU-DAT respectively measured using the evaluation metric explained in section VI-B. The PR values and the percentage accuracy are represented by bars with a scale on the y-axis. We conduct experiments with various confidence thresholds ranging from 0.8 to 0.95 as shown in x-axis. It can be seen, the confidence threshold (δ) of 0.8 results in a higher accuracy of around 90% for both datasets, with precision and recall values of 96% and 92%, respectively. The performance of our proposed system for the other two datasets, CADP and AI CITY Challenge, have an accuracy of around 90%, with precision-recall values of 95% and 91%, respectively. When the confidence threshold is set to be greater than or less than 0.8, an increase in the number of FPs is observed. Since false negatives are more problematic than false positives in this application, a higher recall is more important than a higher precision, and we observe that from the results at all confidence levels.

2) Situation Based ϵ-DDS: To assess our Situation-based DDS's performance in resolving anomalies, we keep track of perturbable clauses in each accident scenario across all four datasets, including our own TU-DAT. As a result, we have 551 perturbable clauses in accident videos from the DAD dataset, 394 clauses in CADP, 225 clauses in our TU-DAT dataset, and 161 perturbable cases in accident videos from the NVIDIA AI City dataset. Fig. 6 shows the results of ϵ-DDS that are averaged over all the four cases as described above. The x-axis indicates the number of iterations needed by combinatorial optimization. It is limited to a maximum of 250, whereas the y-axis shows the percentage of cases where the algorithm could resolve the anomalies. It can be seen that our algorithm, powered by knowledge based on situations or ongoing events, can resolve the anomalies in 100% of the cases in about 310 iterations. In contrast, the normal-DDS can fix only 57% out of the test cases at 250 iterations without any knowledge about the situations or ongoing events.

3) Model's Performance in Terms of Total Execution Time: Additionally, we conducted experiments to see how the C-FAR

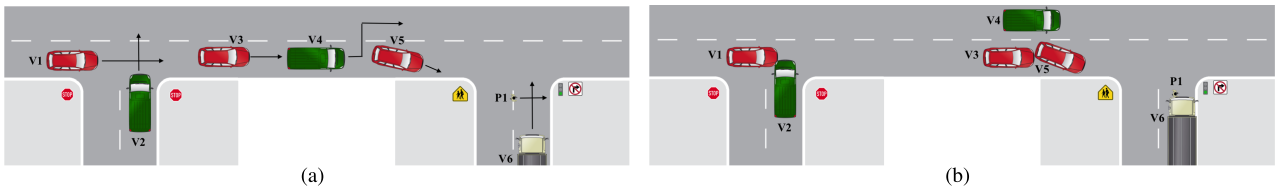

Fig. 4. (a) Initial traffic situation (b) Accident situation.

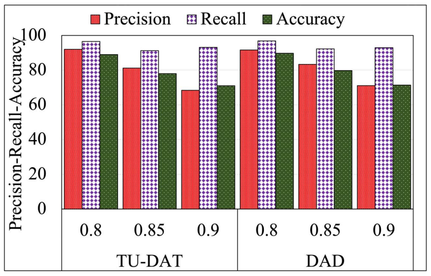

Fig. 5. Accuracy, PR values for DAD and TU-DAT datasets.

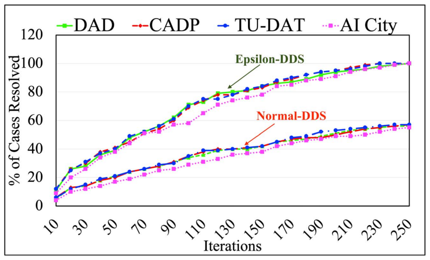

Fig. 6. Performance of situation based ϵ-DDS.

framework's performance varies as the number of events increases. The model receives the events as input, and the time it takes to predict, detect and resolve an anomaly is recorded. These values are compared between number of variables perturbed for $g(x) = |x|$ and $g(x) = x^2$ functions as described in section V-F. It can be seen from Fig. 7 (a) that the time grows slowly at first and then saturates to a linear trend after about 140 events. As a result of the analysis, we also found that, on average, our method can predict an anomaly 4.05 seconds ahead, called *resolution margin time*. Table IV lists the RTM under various accident scenarios. It is evident that the spatio-temporal reasoning system can be easily extended to conduct various investigations of accident scenarios, as the relationship between events and actions can be easily added or modified. Note that 4.05 secs is adequate time for humans to react, and indeed so for the (automated) vehicular safety system (VSS). In fact, the extra time afforded by the proposed approach can be exploited by VSS in two ways: (a) Merge this road-side infrastructure based intelligence with the locally observed situation and make an improved control decision, and (b) apply the desired control gradually and thus more smoothly and with less chance of side effects.

4) Comparison With State-of-Art Methods: We compare our proposed model to three other existing accident detection and prediction models proposed in [3], [31], [32], and the performance results are shown in Fig. 7(b)-(c). In [31], the authors proposed a three-stage framework for auto accident detection in video. The first stage employs a car detection algorithm based on the YOLOv3 deep convolutional neural network; the second stage is a tracking algorithm based on the discriminative correlation filter method, and the final stage employs the Violent Flows (ViF) descriptor to highlight the magnitude changes in the motion vectors that are computed using an optical flow algorithm to detect car crashes. [32] is a framework for detecting anomalies, a 3D neural network architecture based on the EfficientNet 2D classifier for detecting aggressive driving, specifically car drifting. The model proposed in [3] is a Dynamic-Spatial-Attention (DSA) based model that uses RNN along with Long Short-Term Memory (LSTM) cells to model the long-term dependencies of all cues to anticipate accidents in dash-cam videos.

TABLE IV
TABLE: NUMBER OF EVENTS, FLUENTS DEFINED AND RTM FOR VARIOUS ACCIDENT SCENARIOS

Type of Anomaly	#Events	#Fluents	RTM (s)
Pedestrian Accidents	25	17	2.84
Intersection Accidents	37	19	3.42
Rear-end Accidents	41	21	3.85
Front-end Accidents	39	23	3.21
Collision with Static Objects	26	14	4.18
Driving Maneuvers	27	15	3.05
Aggressive Driving	31	18	3.72

We compute the Average Precision (AP) from the sequence of precision and recall pairs, which is used to show the overall accuracy of our C-FAR framework in comparison with other models. From Fig.7 (b) where the x-axis representing the datasets and y-axis representing RTM in seconds, we can observe that our C-FAR framework on the DAD dataset achieves the best AP and best value of Resolution Time Margin (RTM) on the AI city Challenge dataset as shown in Fig.7 (c) where x-axis representing the datasets and y-axis representing AP, which means the model anticipates on average 3.42 seconds earlier before an accident happens while keeping the competitive performance of AP value at 89.27% compared with other three methods. The major disadvantage of the method that combines YOLOv3 and SVM is that the SVM requires several parameters to be set correctly to classify a car crash. Additionally, because the ViF descriptor cannot capture potentially significant orientation changes, the model seems unable to detect multiple objects in a single frame due to interference. For using the DSA+LSTM model, we use the candidate objects and the corresponding CNN features for all the videos in other datasets for a fair comparison. This model requires a significant amount of time and resources to train and prepare for real-world applications. Thus, the DSA+LSTM model becomes highly inefficient from a hardware perspective. To compare our C-FAR framework to the DriftNet architecture, we extracted features and captured the pattern of sliding vehicles using the pretrained EfficientNet3D-B0 model. Pre-training on one or more larger-scale datasets is required when using the DriftNet architecture as a classifier to classify anomalies. Also, the work in [33] has shown that 3D

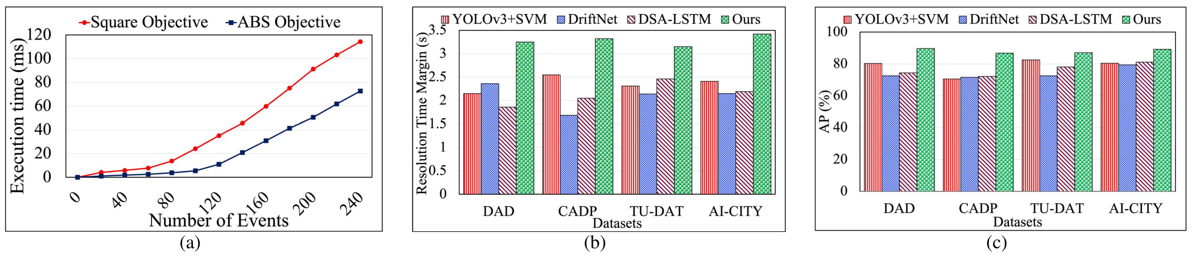

Fig. 7. (a) Total execution time required by C-FAR framework on TU_DAT, (b) and (c): Performance results showing RTM and AP Values for all Datasets.

convolutional networks do not learn efficient representations of videos. The authors have only captured the car drifting behavior in this work. In contrast, C-FAR captures some more aggressive driving behaviors such as speeding, weaving in and out of traffic, and so on while maintaining the highest AP compared to the DriftNet model. Additionally, we compare our C-FAR framework to the DriftNet dataset and observe that C-FAR achieves an accuracy of 95% compared to the DriftNet model's accuracy of 92.5%.

VII. Conclusion

We propose C-FAR framework for predicting and resolving anomalies in intelligent transportation systems using a spatio-temporal event calculus in conjunction with a deep learning model. To test the proposed mechanisms, we harvested many videos of accidents from YouTube. We also generated other videos using a game simulator. A detailed evaluation shows that C-FAR consistently outperforms existing methods in all cases. In particular, C-FAR can predict anomalies with over 90% accuracy at 80% confidence level approximately 4 seconds before the potential accident. Yet the recall rate of the mechanism remains relatively high throughout (about 85% at 90% confidence). In contrast, the current methods only provide an accuracy of around 80%. These mechanisms can be adapted to many other cyber-physical environments such as hospitals, senior care centers, factory environments, etc., which we plan to explore in the future.

References

[1] T. Litman, *Autonomous Vehicle Implementation Predictions*. Victoria, BC, Canada: Victoria Transport Policy Institute, 2017.

[2] A. Artikis *et al.*, "A logic programming approach to activity recognition," in *Proc. 2nd ACM Int. Workshop Events Multimedia*, 2010, pp. 3–8.

[3] F. Chan *et al.*, "Anticipating accidents in dashcam videos," in *Proc. Asian Conf. CV* Cham, Switzerland: Springer, 2016, pp. 136–153.

[4] T. Suzuki *et al.*, "Anticipating traffic accidents with adaptive loss and large-scale incident DB," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 3521–3529.

[5] S. Bouindour *et al.*, "Enhanced CNNs for abnormal event detection in video streams," in *Proc. IEEE 2nd AIKE*, 2019, pp. 172–178.

[6] H. Nguyen *et al.*, "DL methods in transportation domain: A review," *IET ITS*, vol. 12, no. 9, pp. 998–1004, 2018.

[7] U. Ali *et al.*, "Using DL to predict short term traffic flow: A systematic literature review," in *Proc. ITSC*. Cham, Switzerland: Springer, 2017, pp. 90–101.

[8] S. Zhu *et al.*, "Video anomaly detection for smart surveillance," 2020, *arXiv:2004.00222*.

[9] N. Moustafa *et al.*, "Autonomous detection of malicious events using machine learning models in drone networks," in *Proc. 2nd ACM MobiCom Workshop Drone Assist. Wireless Commun. 5G Beyond*, Sep. 2020, pp. 61–66.

[10] I. A. Khan *et al.*, "An enhanced multi-stage deep learning framework for detecting malicious activities from autonomous vehicles," *IEEE Trans. Intell. Transp. Syst.*, early access, Aug. 20, 2021, doi: 10.1109/TITS.2021.3105834.

[11] Y. Wu *et al.*, "A hybrid deep learning based traffic flow prediction method and its understanding," *Transp. Res. C, Emerg. Technol.*, vol. 90, pp. 166–180, May 2018.

[12] J. Wang *et al.*, "Traffic speed prediction and congestion source exploration: A deep learning method," in *Proc. IEEE 16th Int. Conf. Data Mining (ICDM)*, Dec. 2016, pp. 499–508.

[13] P. Pradeepkumar *et al.*, "Resource efficient edge computing infrastructure for video surveillance," *IEEE Trans. Sustain. Comput.*, early access, Mar. 8, 2021, doi: 10.1109/TSUSC.2021.3064245.

[14] E. Letier *et al.*, "Fluent temporal logic for discrete-time event-based models," in *Proc. 10th Eur. Softw. Eng. Conf., 13th ACM SIGSOFT Int. Symp. Found. Softw. Eng.*, 2005, pp. 70–79.

[15] G. Regis *et al.*, "Specifying event-based systems with a counting fluent temporal logic," in *Proc. IEEE/ACM 37th IEEE Int. Conf. Softw. Eng.*, May 2015, pp. 733–743.

[16] I. Chisalita *et al.*, "Traffic accidents modeling and analysis using temporal reasoning," in *Proc. 7th Int. IEEE Conf. Intell. Transp. Syst.*, Jun. 2004, pp. 378–383.

[17] F. Pirri and R. Reiter, "Some contributions to the metatheory of the situation calculus," *J. ACM*, vol. 46, no. 3, pp. 325–361, May 1999.

[18] C. F. Daganzo and N. Geroliminis, "An analytical approximation for the macroscopic fundamental diagram of urban traffic," *Transp. Res. B, Methodol.*, vol. 42, no. 9, pp. 771–781, Nov. 2008.

[19] A. Artikis *et al.*, "An event calculus for event recognition," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 4, pp. 895–908, Apr. 2015.

[20] (2020). *Maxsat 2020*. [Online]. Available: https://maxsat-evaluations.github.io/2020/

[21] B. Tolson *et al.*, "Dynamically dimensioned search algorithm for computationally efficient watershed model calibration," *Water Resour. Res.*, vol. 43, no. 1, pp. 1–16, 2007.

[22] T. Takahama *et al.*, "Constrained optimization by the epsilon constrained hybrid algorithm of particle swarm optimization and genetic algorithm," in *Proc. AI*, 2005, pp. 389–400.

[23] P. Pradeep *et al.*, "Automating conflict detection and mitigation in large-scale IoT systems," in *Proc. IEEE/ACM 21st Int. Symp. Cluster, Cloud Internet Comput. (CCGrid)*, May 2021, pp. 535–544.

[24] C. Ansótegui *et al.*, "A new algorithm for weighted partial MaxSAT," in *Proc. AAAI Conf. AI*, 2010, vol. 24, no. 1. [Online]. Available: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.231.4748&rep=rep1&type=pdf

[25] *Physics Tutorial*. Accessed: Apr. 6, 2021. [Online]. Available: https://www.physicsclassroom.com/class

[26] A. Shah *et al.*, "CADP: A novel dataset for CCTV traffic camera based accident analysis," 2018, *arXiv:1809.05782*.

[27] M. Naphade *et al.*, "The 5th AI city challenge," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2021, pp. 4263–4273.

[28] M. Bortnikov *et al.*, "Accident recognition via 3D CNNs for automated traffic monitoring in smart cities," in *Proc. SAI Conf.* Cham, Switzerland: Springer, 2019, pp. 256–264.

[29] *Beamng.drive*. Accessed: Sep. 15, 2021. [Online]. Available: https://www.beamng.com/game/

[30] *CVAT*. Accessed: Sep. 1, 2021. [Online]. Available: https://github.com/openvinotoolkit/cvat

[31] V. M. Arceda and E. L. Riveros, "Fast car crash detection in video," in *Proc. 44th Latin Amer. Comput. Conf. (CLEI)*, Oct. 2018, pp. 632–637.

[32] A. Noor *et al.*, "DriftNet: Aggressive driving behaviour detection using 3D CNNs," in *Proc. SMARTTECH*, 2020, pp. 214–219.

[33] D. Tran *et al.*, "Learning spatiotemporal features with 3D convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 4489–4497.