Neuro-symbolic Complex Event Recognition in Autonomous Driving

Tatiana Boura[a,*] **and Nikos Katzouris**[a]

[a]Institute of Informatics and Telecommunications, NCSR ‘Demokritos’, Ag. Paraskevi, Greece

Abstract. Complex Event Recognition (CER) systems aim to identify critical events of interest that emerge from streams of data. These complex events are typically specified as spatio-temporal compositions of simpler events (e.g., sensor readings), using symbolic temporal patterns such as finite state machines or temporal logic rules. In practice, CER systems often need to operate on sub-symbolic input. For instance, autonomous vehicles must detect complex, temporally extended events, such as overtaking maneuvers, based on raw sensor data like video streams, in order to react safely and effectively. Neuro-symbolic (NeSy) AI offers a promising framework in this context, as it combines neural networks’ ability to interpret sub-symbolic data with symbolic reasoning over structured knowledge, such as CER patterns. However, the application of NeSy techniques to temporal learning and reasoning in real-world domains remains significantly underexplored. To address this gap, we propose a NeSy approach, which utilizes the NeSyA framework, for detecting overtaking events between vehicles in an autonomous driving setting. We conduct an empirical evaluation on the ROAD dataset and demonstrate that our approach outperforms purely neural baselines in terms of complex event recognition performance.

1 Introduction

Many applications require processing of continuously streaming data from geographically dispersed sources. Complex event recognition (CER) involves identifying events within these streams, enabling the implementation of both reactive and proactive actions [15]. Beyond their time efficiency, CER systems are valued for their emphasis on trustworthy decision-making. This is achieved through well-defined theoretical frameworks, such as logic specifications and automata, and machine learning methods like Inductive Logic Programming and structure learning, which provide symbolic pattern definitions, sound pattern learning and efficient inference.

However, in applications involving sub-symbolic input, such as video data, there is a need to integrate these symbolic methods with sub-symbolic models to maintain performance. This necessity motivates the introduction of Neuro-Symbolic Artificial Intelligence (NeSy) into the CER domain. NeSy systems integrate neural-based learning with logic-based reasoning, combining sub-symbolic data processing with symbolic knowledge representation. This integration aims to enhance interpretability, robustness, and generalization of sub-symbolic methods, particularly improving their capacity to handle out-of-distribution data.

A relevant domain for the integration of NeSy methods and CER is autonomous driving, since –given the mission-critical nature of this domain– event recognition must be both efficient and reliable. In this context, vehicles must interpret data from cameras and sensors to quickly identify events that may require action. Many events in this domain can be formally described using rules and enriched with background knowledge, which can be effectively defined and leveraged through CER methods.

In this setting, simple event predictors can be modeled using sub-symbolic structures, while complex event recognition is addressed through established symbolic CER frameworks. Several NeSy works have been proposed that handle temporal dynamics present in data sequences [34, 5, 31, 6, 7], but they are application specific and do not offer a generalized framework that learns over a formalization of simple events. On the other hand, generalizable NeSy frameworks such as DeepStochLog [33], DeepProbLog [24], and NeurASP [35] are not inherently designed to model temporal events and need to be enforced with time-aware reasoning (e.g. timestamps, sequential neural models, stochastic processes etc.). One model that addresses both limitations is NeSyA (Neuro-Symbolic Automata) [23], which combines symbolic automata with neural-based perception under probabilistic semantics in an end-to-end differentiable framework. NeSyA supports temporal reasoning while enabling the learning of common symbolic structures used in CER.

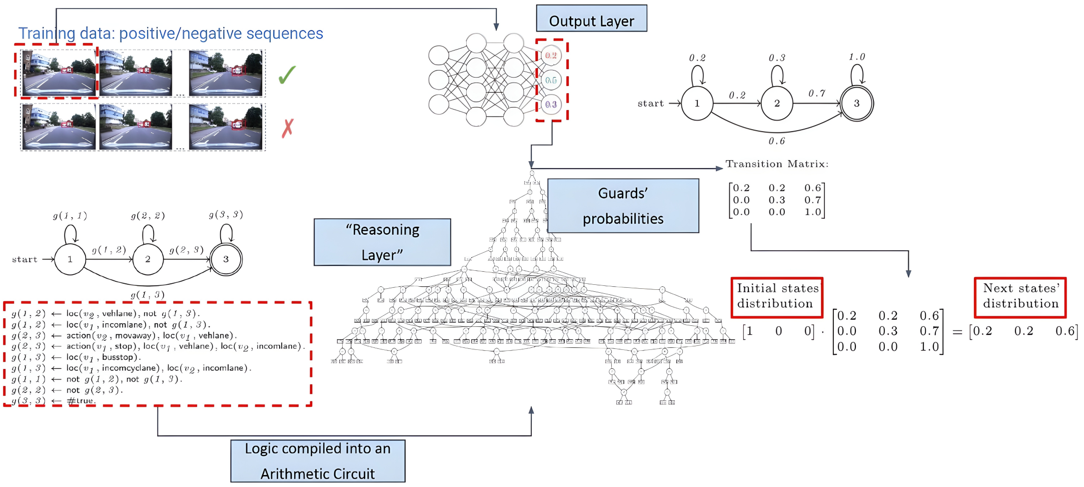
This work represents an initial effort to address complex events in autonomous driving with NeSy, and specifically NeSyA, with the incentive to yield better results than purely neural approaches, focusing on the recognition of overtake incidents between agents in the ROAD dataset [29]. The remainder of the paper is structured as follows. Section 2 presents the necessary theoretical background, focusing on CER and its relation to symbolic automata and autonomous driving. Section 3 outlines our neuro-symbolic approach, explaining the integration of the sub-symbolic models and symbolic automata for training and inference. The complete dataset, the experimental setup, results and analysis are provided in Section 4, for the challenging task of recognizing the complex event where a road agent overtakes another. Finally, Section 5 concludes the paper and outlines directions for future work.

2 Background

2.1 Complex Event Recognition

Complex Event Recognition (CER), also known as complex event pattern matching, refers to the detection of complex events in streaming data by identifying temporal patterns composed of simple events,

[*] Corresponding Author. Email: tatianabou@iit.demokritos.gr.


Training data: positive/negative sequences
✓
✗
Output Layer
0.2
0.5
0.3
start
1
2
3
0.2
0.3
1.0
0.2
0.7
0.6
Transition Matrix:
$\begin{bmatrix}0.2 & 0.2 & 0.6\\ 0.0 & 0.3 & 0.7\\ 0.0 & 0.0 & 1.0\end{bmatrix}$
Guards' probabilities
"Reasoning Layer"
g(1, 1)
g(2, 2)
g(3, 3)
g(1, 2)
g(2, 3)
g(1, 3)
Initial states distribution
Next states' distribution
$[1 \quad 0 \quad 0] \cdot \begin{bmatrix}0.2 & 0.2 & 0.6\\ 0.0 & 0.3 & 0.7\\ 0.0 & 0.0 & 1.0\end{bmatrix} = [0.2 \quad 0.2 \quad 0.6]$
Logic compiled into an Arithmetic Circuit

```
g(1, 2) ← loc(v2, vehlane), not g(1, 3).
g(1, 2) ← loc(v1, incomlane), not g(1, 3).
g(2, 3) ← action(v2, movaway), loc(v1, vehlane).
g(2, 3) ← action(v1, stop), loc(v1, vehlane), loc(v2, incomlane).
g(1, 3) ← loc(v1, busstop).
g(1, 3) ← loc(v1, incomcyclane), loc(v2, incomlane).
g(1, 1) ← not g(1, 2), not g(1, 3).
g(2, 2) ← not g(2, 3).
g(3, 3) ← #true.
```

Figure 1. Illustration of the inference and training procedure in NeSy-SFA. First, videos are processed through a neural network that outputs simple event probability distributions (*Output Layer*). These probabilities help answer the probabilistic query of whether the sequence is in a certain state at a given frame (*Guards' Probabilities*), utilizing a compiled Boolean circuit (*Logic compiled into an Arithmetic Circuit*). Over time, each state accumulates probabilities, by multiplying the probability distribution with the *Transition Matrix*, resulting in a final state probability distribution at the end of the sequence. We use this distribution to compute the loss for the 'overtake' incident prediction and backpropagate the loss to train the network, repeating the process until we achieve the minimum loss value.

i.e. low-level occurrences, or even other complex events [8]. Typically, CER systems operate on streams of event tuples [15, 16], which are time-stamped collections of attribute-value pairs. Conceptually, CER input can be seen as a multivariate sequence, with one subsequence per event attribute. For example, an attribute might represent the output of a specific sensor, and its values correspond to the sensor's readings over time, whether numerical, categorical, and/or sub-symbolic. Each event tuple serves as a observation of the joint evolution of all relevant attributes at a specific time point. Complex event patterns define both a temporal structure over these event tuples and a set of constraints on their attributes. A pattern is matched when a sequence of event tuples satisfies both the required temporal ordering and the attribute constraints.

These patterns are typically specified by domain experts using event specification languages [16]. Such languages must support a core set of event-processing operators [2, 15, 37], including: (a) sequence, indicating that specific events must occur in temporal succession; (b) iteration (Kleene Closure), requiring one or more repeated occurrences of an event type; (c) filtering, which restricts matches to events satisfying predefined predicates.

These operators naturally align with a computational model based on Symbolic Finite Automata (SFAs) [12]. Unlike classical automata, which assume finite alphabets, SFAs generalize transitions to be governed by logical predicates over potentially infinite domains, represented using effective Boolean algebras [32, 30]. This enables expressive and compact representations of complex event structures. As a result, most existing CER systems rely on SFA-based pattern representations [1, 37, 17, 3, 11, 4, 10]. In these systems, patterns are typically written in declarative languages (e.g., SQL-like syntax) and compiled into symbolic (often non-deterministic) automata.

2.2 *CER in Autonomous Driving*

Existing work in the autonomous driving domain typically describes activities as driving events, i.e., events occurring during driving [25, 21, 36]. The connection to the CER theory is evident: autonomous vehicles must process numerical and/or sub-symbolic sensor data to recognize driving events. For example, sudden braking may follow a sequence in which a car stops at a red light, accelerates when it turns green, and then brakes abruptly as a deer crosses the road.

Framing these problems as CER tasks is motivated by the fact that many target patterns are either known or can be explicitly defined. When such patterns are not predefined, learning-based methods can be used to discover patterns compatible with CER systems. A relevant example is the ROAD dataset [29, 28], a richly annotated autonomous driving dataset based on the RobotCar dataset [22]. ROAD provides frame-level annotations for agents, including their identity (e.g., vehicle, pedestrian), action(s) (e.g., overtaking, turning left), and semantic location(s) (e.g., left pavement, incoming lane).

From a CER perspective, certain actions, such as 'overtake', constitute complex events, while others, such as 'green traffic light', represent states. Among these, 'overtake' is particularly notable due to its temporal extent, involvement of multiple (simple) sub-events, and significant impact on the scene, making it a compelling CER task. However, the 'overtake' pattern is not predefined. Given the complexity of scenes-multiple agents, dynamic locations, and concurrent actions, and the lack of domain experts, manual specification is infeasible. Section 4 details the learning approach used to extract such patterns.

3 Neuro-Symbolic Approach

To perform CER on the video input, we combine ideas from (sequential) NeSy frameworks and standard CER pipelines: neural networks process sub-symbolic input to detect simple events (actions and semantic locations), while symbolic automata handle pattern matching to recognize 'overtake' incidents. In this Section we will describe in detail the NeSy integration in our work, by outlining the NeSyA framework and its theoretical basis and connecting it to our decisions, driven by the task at hand.

Table 1. Labels and locations for agents, along with available actions for both agents and the autonomous vehicle.

Agent Labels	AV Actions	Agent Actions	Agent Locations
Pedestrian	Stop	Move away, towards	AV lane
Motorbike	Move	Move right, left	Outgoing lane
Bus	Turn right,left	Move	Outgoing cycle lane
Car	Move right, left	Brake	Incoming lane
Medium vehicle	Overtake	Stop	Incoming cycle lane
Large vehicle		Indicating left, right	Pavement
Cyclist		Hazard lights on	Left pavement
AV traffic light		Turn left, right	Right pavement
Other traffic light		Push object	Junction
Emergency vehicle		Reversing	Crossing location
		Overtake	Parking
		Red, Green, Amber light	Bus stop
		Wait to cross / Crossing / Cross from left, right	

3.1 SFAs and Markov Models

In sequence modeling, it is often reasonable to assume that recent observations are more predictive than distant ones. This motivates the use of Markov models, where future states depend only on a limited history, typically just the current or previous state [9]. In these models, transitions between states are governed by probabilities. A model is considered non-stationary if these transition probabilities change over time.

Markov models represent sequences using a state space and a transition function in the form of a matrix that defines the likelihood of moving from one state to another. At each time step, the distribution over states is updated based on the previous distribution and the current transition probabilities. This formulation allows for efficient modeling of temporal dynamics in data.

A seemingly different approach comes from SFAs. Rather than using probabilities, SFAs define transitions using logical conditions over structured inputs. Specifically, inputs are interpreted as truth assignments over a set of propositional variables and transitions occur when the current input satisfies a logical formula attached to an edge in the automaton.

Both frameworks process sequences by transitioning through states in response to observed inputs, whether those inputs are numeric symbols or logical interpretations and when SFAs are applied to data streams (where input patterns or variable co-occurrences can be estimated) transitions can be interpreted probabilistically, much like in a non-stationary Markov chain. So, SFAs can subsume Markov models by encoding structured dependencies while remaining amenable to probabilistic analysis.

3.2 Differentiable Probabilistic Inference via SFAs

Probabilistic reasoning over structured domains typically involves modeling uncertainty using joint probability distributions over finite sets of variables [14, 26]. While expressive, these distributions grow exponentially with the number of variables, rendering exact inference intractable. A widely used approach to address this is Weighted Model Counting (WMC), which encodes the probabilistic model as a weighted logical theory, consisting of a propositional formula and a function assigning weights (probabilities) to literals [27]. The probability of a query is then computed by summing the weights of all satisfying assignments, generalizing the classical model counting problem.

This process underlies probabilistic logical inference, where one computes the probability that a logical formula holds under uncertain inputs. Since WMC is a $\#P$-complete problem, practical inference relies on Knowledge Compilation, which transforms formulas into tractable representations, such as deterministic decomposable negation normal form (d-DNNF) circuits [13]. Once compiled, inference becomes linear in the size of the circuit and differentiable.

Symbolic automata define transitions between states using propositional formulas over input variables. When inputs are uncertain or noisy, each transition can be evaluated probabilistically by applying WMC to the corresponding formula. If the automaton is constructed using compiled circuits for each transition, the entire system becomes a differentiable probabilistic model, enabling integration with gradient-based learning methods.

3.3 End-to-end training

Let us present in this section the NeSy pipeline in both inference and learning scenarios. Note that the process of probabilistic inference and learning is embedded in NeSyA, but we will not distinguish it here so that the pipeline is more coherent. We begin by outlining the inference process –a single feed-forward pass from input to prediction.

A video is processed by simple event recognition networks, which output probability distributions over simple events, specifically each two agents' actions and semantic locations for every frame. These distributions are then used to classify (ground) the agents discrete actions and locations for the evaluation of the symbolic automaton. Next, a smooth d-DNNF circuit is compiled from the ASP representation of the automaton. The circuit includes one variable for each possible action and location value, and supports probabilistic queries corresponding to the automatons transitions. These queries form the transition matrix by computing weighted model counts that accumulate probabilities in the states of the automaton.

For each video, a row vector representing the probability distribution over automaton states at each time step is maintained. It is initialized such that the start state has probability mass 1, with all others set to 0. As each frame is processed, the state vector is updated by multiplying it with the current transition matrix. Each column of the transition matrix represents the probability of transitioning into a particular state at a given frame. Because the transition matrix is computed from neural network outputs, which vary at every timestep, we consider our symbolic automata non-stationary. The final output is the state distribution after processing the last frame.

We now turn to the learning procedure. After each forward pass, the computed state probability distribution can be used to evaluate the prediction loss over the complex event. This loss can be defined over the entire distribution or based solely on the acceptance probability –that is, the probability mass assigned to the automatons final (accepting) state. Since the compiled symbolic automaton is differentiable, the loss can be backpropagated through the symbolic layer. This enables end-to-end training of the simple event recognition networks via gradient descent. As a result, the model learns to adjust its predictions of simple events in a way that improves recognition of complex events, which in our task is the 'overtake' event through

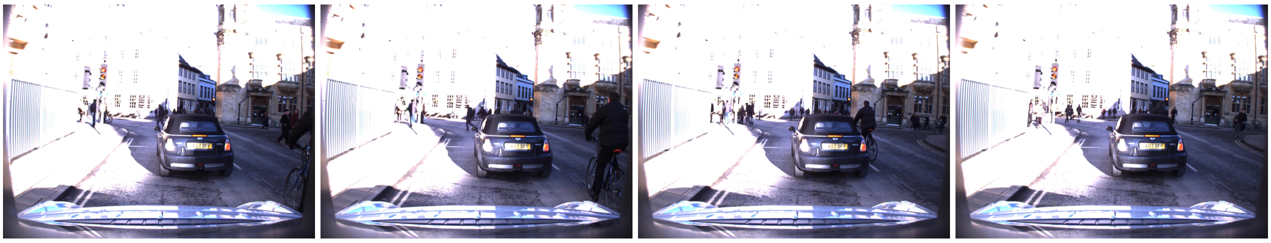


Figure 2. A bicycle approaches from behind the AV, overtakes it while the AV moves forward, and stops at a red light. It then continues to overtake a car that is stopped ahead of the AV at the traffic light.

distant supervision. A visualization of the proposed pipeline is presented in Figure 1.

4 Experiments

4.1 Sequential datasets

ROAD dataset consists of 22 real-world 8-minute videos recorded between November 2014 and December 2015 in central Oxford, covering a range of routes and seasonal conditions. Of these, 20 videos are currently available for training and evaluation.

Road events are defined as a series of bounding boxes linked in time (frames), annotated with the agents label, action(s), and semantic location(s) (cf. Table 1). Regarding the autonomous vehicle (AV), we only know its unique ego-action (Table 1). Each agent has a unique identifier per video. The dataset includes approximately 122K annotated frames (12 fps) at 1280 × 960 resolution with a multitude of agents per frame.

Regarding the complex 'overtake' actions, the dataset contains 30 unique overtakes, performed either by the AV or other agents. Durations range from 2 to 164 frames (mean: 49.83; std: 41.87), all occurring within 9 videos. Figure 2 illustrates an overtaking instance from the ROAD dataset.

To enable neurosymbolic integration and construct a pipeline that extracts sub-symbolic information from video and feeds it into a symbolic reasoning module for overtake recognition, we extract two aligned sequential datasets from the complete dataset: one symbolic and one sub-symbolic, in one-to-one correspondence. We differentiate between overtakes involving the AV and those involving two external agents. This distinction is necessary, as each type exhibits different visual and symbolic patterns. When the AV is involved, its position is fixed, and its visual representation is not relevant, unlike scenarios where the AV is not part of the overtake. The dataset consists of sequences ranging from 6 to 10 frames (approximately 0.5 to 1 second), a duration sufficient for humans to recognize overtakes in both symbolic and sub-symbolic modalities.

We define three classes: 0 for negative examples (no overtake), 1 when the first agent overtakes the second, and 2 when the second agent overtakes the first. This labeling explicitly captures the directionality of the overtake. Positive instances were generated by selecting video segments with a maximum length of 10 frames, using non-overlapping chunks to prevent overfitting during NeSy training. A sliding window approach was avoided due to the limited number of positive examples, which would result in highly similar instances. This process yielded 92 positive instances, each concluding with and containing an overtake event.

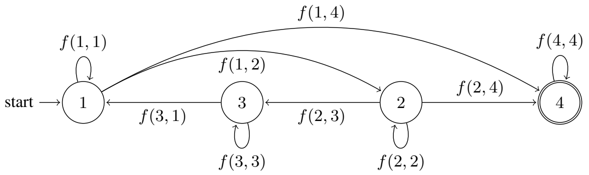
Selecting negative instances is inherently more challenging, as any sequence not classified as an overtake could theoretically serve as a negative. To ensure informative training, we focused on close negatives: sequences that initially resemble overtakes but do not culminate in one vehicle passing another. To construct these, we identified the action pairs performed by agents prior to overtakes, along with their frequency, and stochastically searched the dataset for similar sequences that do not result in overtakes. Only one instance per agent pair was included, and both agents were required to appear for at least 6 frames. This process yielded approximately 2,000 negative instances. While downsampling negative examples could balance the dataset, we deliberately avoided this approach. Overtake events are inherently sparse, and artificially balancing the dataset would introduce unrealistic conditions. Also training on simplified, artificially balanced data would lead to poor performance, given the sub-symbolic complexity of the task.

The *symbolic* dataset provides a structured, logic-based representation of events occurring within each frame. Each instance encodes facts describing the two agents involved, including their identity (e.g., AV, large vehicle), actions, semantic locations, and normalized bounding box coordinates at each timepoint (frame). The instance's class label is also included. This representation enables the grounding of the complex overtake event in terms of simple events, defined by combinations of agent actions and locations within the symbolic framework. The *sub-symbolic* includes the corresponding images of the frames that consist the symbolic dataset.

To ensure unbiased evaluation, we enforced a strict separation between training and testing sets, preventing overlap of augmented positives or negatives from the same video segments. We performed an 80/20 train/test split, analogous to k-fold cross-validation, using disjoint sets of videos for positive samples. This resulted in up to 36 splits, allowing testing on out-of-distribution data. While we initially applied the same strategy to negative samples, we observed a drawback: videos vary significantly in visual characteristics (e.g., snow-covered vs. leafy junctions), and training solely on one type reduces generalization. To mitigate this, we allowed negatives from all videos but enforced a minimum temporal distance of 100 frames between any two selected instances, avoiding redundancy while maintaining visual diversity.

To simplify the task, we focused only on one positive class and overtakes not involving the AV. As a result, not all data splits remained suitable, since some lacked relevant positives or exhibited more positives in the testing set. We randomly selected four viable splits for training and evaluation. Across these splits, the number of positive sequences in the training set ranges from 46 to 75, and from 17 to 46 in the test set. The corresponding number of negative sequences is approximately 550 for training and 250 for testing.

4.2 Extracting Background Knowledge



```
% State 1 -> 2: if agent 2 is moving towards
% the AV and not transitioning to State 4.
f(1,2) :- action_2(movtow), not f(1,4).
% State 1 -> 4: if agents are in the same lane
% and the agent 2 is moving away from the AV.
f(1,4) :- same_lane(l1, l2), action_2(movaway).
% Stay in State 1: if not moving to State 2 or 4.
f(1,1) :- not f(1,2), not f(1,4).

% State 2 -> 3: if agent 2 is moving towards
% the AV and not transitioning to State 4.
f(2,3) :- action_2(movtow), not f(2,4).
% State 2 -> 4: if agent 1 is stopped and
% agent 2 is in the incoming lane.
f(2,4) :- action_1(stop), location_2(incomlane).
% Stay in State 2: if not moving to State 3 or 4.
f(2,2) :- not f(2,3), not f(2,4).

% State 3 -> 1: if agent 1 is in the incoming lane.
f(3,1) :- location_1(incomlane).
% Stay in State 3: if not moving to State 1.
f(3,3) :- not f(3,1).

% Stay in State 4: always; absorbing state.
f(4,4) :- #true.
```

Figure 3. Learned automaton from symbolic dataset. `l1` and `l2` denote the agents' (with the respective index) locations. It achieves an *F1-score* of approximately 0.87 on the test set. The actual ASP syntax has been simplified for clarity of the illustration.

Since 'overtake' patterns were not predefined, we employed the ASAL framework [19] to learn the patterns from the symbolic sequential dataset. ASAL learns Answer Set Automata, an extension of SFAs tailored for CER over multivariate event streams, where transition predicates are defined via ASP rules. Through declarative learning with symbolic reasoning it produces compact models with strong generalization performance.

We used ASAL with the objective of maximizing generalization on the test set. We learned a general automaton from the different symbolic splits. This led to the selection of a subset of simple events most relevant for complex event recognition. The selected actions were: *moving away*, *moving towards*, *stop*, and *other* (none of the above). The selected semantic locations were: *incoming lane*, *vehicle lane*, *junction*, and *other*. Intuitively, this aligns with human reasoning: recognizing an 'overtake' primarily requires understanding the orientation and motion direction of the vehicle.

The above process resulted in the automaton shown in Figure 3. This learned symbolic automaton accepts multiple patterns as valid instances of overtakes, represented by different paths leading to the accepting state. Examples of such paths include: `f(1,1)` → `f(1,1)` → `f(1,2)` → `f(2,4)` or `f(1,1)` → `f(1,4)`. Let us give an intuitive overtaking pattern that is validated by the shortest accepting path `f(1,4)`:

- AV detects two vehicles in the same lane as itself (vehicle lane)
- Both vehicles are visible in front of the AV, meaning they are positioned side by side without overlapping in the AV's field of view
- If one of these vehicles is detected as moving, while the other is static or moving slower, the moving vehicle is classified as overtaking the other

4.3 Experimental Setup

In a higher level of abstraction, the task is framed as a binary sequence classification problem: determining whether a given sequence of frames constitutes an 'overtake'. Experiments were conducted on the four (sub-symbolic) data splits described in Section 4.1. We trained NeSy models and compared their performance against purely neural baselines.

For simple event recognition, we employed two architectures: a 2D-CNN for semantic location prediction and a 3D-CNN for action recognition, both with multiple convolutional layers. The temporal modeling capability of the 3D-CNN is particularly important for recognizing motion-based actions. Each module outputs eight predictions per frame: probability distributions over the actions and locations of each agent. Although the annotations are multi-label (e.g., an agent may simultaneously move toward the AV and signal a left turn), the task is cast as multi-class due to the requirement in the NeSy pipeline for probability distributions over mutually exclusive classes. Both networks receive the same input: a 10-frame video segment and bounding boxes of the two agents of interest per frame.

To evaluate the temporal reasoning capabilities of our NeSy model, we compare it against a standard spatio-temporal neural architecture: a Long Short-Term Memory (LSTM) network [18]. In this baseline, the outputs of the simple event recognition modules are passed to an LSTM (hidden size 10), whose output is used to predict the final classification probability.

For training, we used the Adam optimizer [20] with a batch size of 8. Due to the differing temporal context –80 frames for the semantic location network versus 8 for the action recognition network– we set distinct learning rates for each. Empirically, we found that the semantic location module required a lower learning rate, so we used 10^{-5} for the 3D-CNN action recognizer and halved it for the location module.

All CER models were trained for a fixed 40 epochs. The neural baseline took approximately 20 seconds per epoch, whereas for the NeSy approach took 30 seconds. Given the scarcity of positive examples in the training set, we did not employ a validation set. Instead, model selection was based on training loss dynamics: we normalized losses to the $[0, 1]$ range using the first epoch's loss as the maximum and 0 as the minimum, then selected the model at the earliest epoch where the loss plateaued, defined as a change of less than 0.05 across a window of two consecutive epochs.

Since 'overtake' instances are sparse, comprising only 10% of the dataset, the task becomes a highly imbalanced binary classification problem. To address this, we evaluated two loss functions for NeSy and baseline training: *weighted binary cross-entropy* (weighted BCE) and *focal loss*. While weighted BCE increases the contribution of the minority class by reweighting class loss terms, focal loss down-weights easy examples, focusing learning on harder, misclassified ones.

In the neural baseline, outputting a complex event probability is straightforward. In contrast, the NeSy model produces a state probability distribution over the automaton. The first and last entries in this vector correspond to the start and accepting states, respectively. We experimented with two approaches for mapping this distribution to a classification probability: (a) using only the acceptance probability, and (b) comparing the full state distribution to the target distribution $(0, 0, 0, 1)$ using the Kolmogorov-Smirnov (KS) distance. The KS distance provides a bounded $[0, 1]$ similarity score between cumulative distributions, offering a principled, interpretable metric to evaluate whether the final state is reached.

4.4 *Results and Discussion*

4.4.1 *End-to-end NeSy*

Our primary objective is to evaluate complex event recognition, i.e., the recognition of the 'overtake' event, across the four sub-symbolic data splits. To ensure a fair comparison across splits with imbalanced class distributions, we adopt the micro-averaged F1 score as our evaluation metric across all data splits. Table 2 presents the comparative results on complex events for the NeSy and baseline for all loss configurations.

Table 2. Micro-F1 scores by model type, loss function, and NeSy probability variant. 'States' uses the full state distribution; 'Final' uses only the acceptance probability. A random 50% predictor yields 0.13 micro-F1.

Metric	Baseline		NeSy			
	Focal	Weighted BCE	Focal		Weighted BCE	
			States	*Final*	*States*	*Final*
Micro F1	0.15	0.14	**0.55**	0.42	0.31	0.39

Overall, the NeSy counterpart outperforms the neural baseline by a large margin across all configurations. Additionally, focal loss yields better performance than weighted BCE in both model types. However, no single acceptance probability computation strategy consistently outperforms the other across all loss types within the NeSy configurations. Specifically, using the full state probability distribution is superior when employing focal loss, whereas relying solely on the acceptance probability yields better results under weighted BCE.

This discrepancy can be attributed to the characteristics of each loss function. Focal loss is particularly effective at emphasizing hard, misclassified examples, especially from the minority class. In such cases, the richer information provided by the full automaton state distribution enables finer-grained adjustments that help reduce loss more effectively. The KS-derived score, computed from the full distribution, provides a softer, less confident prediction signal that is less biased and better reflects uncertainty across states. Focal loss benefits from this nuance, as it is designed not for probability calibration but for modulating loss based on prediction confidence. In contrast, weighted BCE operates as a weighted maximum likelihood estimator under asymmetric class priors, assuming calibrated, true probabilities as input. Consequently, it performs best when provided with a single, well-defined probability –such as the acceptance probability– rather than a heuristic proxy derived from distributional similarity.

4.4.2 *Evaluation on Simple Events*

However, as seen in Table 2, the F1 scores on the testing set remain relatively low. Again, as mentioned in Section 4.1, the computer vision task itself is difficult, so low scores in the distant supervision task of classifying an 'overtake' is expected. Additionally, for the neural baseline, this outcome is expected due to the high variability among 'overtake' instances, which hinders generalization. In contrast, the reduced performance of the NeSy model suggests deficiencies in simple event recognition, since the symbolic automaton, demonstrates high generalization on the testing set.

To investigate this hypothesis, we overfit a NeSy model on the training set and then evaluate its simple event predictors directly on the training data. As shown in Table 3, although the model achieves perfect recognition of 'overtake' instances, it relies on what can be described as reasoning shortcuts: it learns to exploit superficial cues in the input to satisfy the automaton transitions without truly understanding or modeling the intended semantics of the simple events. Note that in preliminary experiments we also used pre-trained simple event predictors, but the complex event training still managed to find the best training shortcut.

Table 3. Trained complex event predictor evaluated on simple events for only one split. The model is overfitted on the training set to isolate the symbolic components behavior. Evaluation is reported as per-class F1 scores (one-vs-all) for each simple event category.

Complex Event (F1-Score) - *Training Set*			0.99		
Action			**Location**		
Class	Micro-F1	Support	**Class**	Micro-F1	Support
Move away	0.301	1335	Vehicle Lane	0.000	1321
Move towards	0.273	6752	Incoming Lane	0.137	7194
Stop	0.301	2350	Junction	0.000	1967
Other	0.000	3963	Other	0.687	3918

4.4.3 *Loosely coupled NeSy*

Given the sub-optimal performance of the NeSy model, one natural consideration is to decouple training and reasoning, i.e., to first train the simple event predictors independently, and then incorporate the symbolic component only at inference time.

Two approaches are possible: (a) utilizing the entire dataset for the simple event prediction task, and (b) utilizing only the sub-symbolic dataset splits defined for the end-to-end NeSy task, as described in Section 4.1. The evaluation results for the simple event predictors trained using these two approaches are presented in Table 4. As expected, leveraging a larger portion of the dataset for training leads to improved performance in simple event recognition. However, since our primary evaluation pertains to the NeSy training and inference process, we proceed with the simple event predictors trained on the dataset used for the end-to-end NeSy component. This configuration serves as the baseline for the current task definition and dataset setup.

If we evaluate 'overtake' recognition using the pre-trained simple event recognizers by appending the symbolic automaton, the results show that relying solely on this sequential setup, without end-to-end training, yields a complex event F1 score of 0.0, indicating that end-to-end training is essential for achieving non-trivial performance.

However, while the overall complex event performance is low, a score of exactly zero suggests further investigation. We therefore conduct an additional experiment in Table 5, where we evaluate complex event recognition while selectively fixing some simple event predictions to their ground-truth labels. This allows us to assess whether the accurate prediction of specific simple events has a disproportionately large influence on complex event recognition and whether certain errors in simple event prediction are particularly detrimental.

If we provide the symbolic automaton with the ground-truth distribution of all simple events, as expected, we recover the automaton's maximum F1 score on the testing set (cf. Figure 3). When providing only the ground truth for the agents' actions, the 'overtake' recognition F1 score increases to 0.43. In contrast, supplying only the ground truth for the agents' semantic locations yields a much lower score of 0.01. Interestingly, when fixing agent 1's action and agent 2's location to their true values, the F1 score rises to 0.81, very close to the automaton's upper limit.

This observation highlights that not all simple event predictions contribute equally to complex event recognition. Intuitively, one

Table 4. Simple event training and F1 scores for action and semantic location recognition on the testing set using two training configurations: (a) (left) models trained on a randomly selected 60% of the dataset, validated on 20% with early stopping, and tested on the remaining 20%; (b) (right) models trained on the four sub-symbolic splits used for the complex event task. Actions and locations for both agents are evaluated jointly to keep the table simple.

(a) Trained on the whole dataset

Action		Location	
Class	F1 Score	**Class**	F1 Score
Move away	0.980	Vehicle Lane	0.767
Move towards	0.857	Incoming Lane	0.822
Stop	0.889	Junction	0.905
Other	0.861	Other	0.715

(b) Trained on the complex event dataset

Action		Location	
Class	Micro-F1	**Class**	Micro-F1
Move away	0.328	Vehicle Lane	0.504
Move towards	0.686	Incoming Lane	0.408
Stop	0.573	Junction	0.491
Other	0.158	Other	0.532

Table 5. Results on the loosely coupled NeSy structure. Simple event predictors trained on the NeSy dataset splits are evaluated on the complex event. Some simple events are given their true labels during evaluation. `action_1` refers to agent 1's action, `location_1` to their location, etc.

Fixed Simple Events	Micro F1
None	0.00
All	0.87
`action_1`, `action_2`	0.43
`location_1`, `location_2`	0.01
`action_1`, `location_2`	0.81

might expect that accurate semantic location predictions would significantly improve performance, as predicates such as `same_lane`, `location_1`, and `location_2` appear in multiple transitions within the automaton, but that is not the case. On the contrary, examining the learned automaton reveals that `action_1` is involved only in the transition `f(2,4)`, where it is conjuncted with `location_2(incomlane)`. Accurately predicting this specific conjunction appears to be critical for achieving high complex event recognition performance. These results indicate that certain transitions in the symbolic automaton are more crucial for temporal reasoning than others, and accurate prediction of the literals involved in these key transitions has a disproportionately large impact on overall complex event recognition.

5 Conclusions and Future Work

In this work, we presented a Neuro-Symbolic (NeSy) pipeline for Complex Event Recognition, focusing on the recognition of overtake incidents between two vehicles from video data. Our experiments demonstrate that the NeSy model significantly outperforms its purely neural counterpart across all configurations.

We also evaluated the learned simple events as well as a loosely coupled NeSy setting. Interestingly, our findings show that the end-to-end NeSy model does not rely solely on accurate simple event predictions for correct complex event recognition; instead, it is subject to reasoning shortcuts. In the loosely coupled setting, we observed that the importance of specific simple events depends more on their role in key automaton transitions rather than on their frequency within the automaton structure.

A primary direction for future work is the reformulation and expansion of the dataset. Incorporating more data and a broader range of complex events would address one of the main limitations of our study, namely, the limited training data combined with the inherent complexity of the computer vision tasks involved.

Another promising direction is the systematic study of the relationship between symbolic automaton structure and NeSy training dynamics. It is plausible that certain automaton architectures are more suitable for guiding the neural component. For instance, automata with fewer conjunctive conditions in their transitions may make the simple event training easier, while more complex automata could offer smoother convergence or improved generalization.

Finally, a highly relevant avenue is the joint learning of both the neural and symbolic components. Instead of fixing background knowledge in advance, we could provide a flexible knowledge base and allow the system to learn both the automaton structure and the neural network parameters simultaneously. While this approach poses considerable challenges, it holds the potential for creating more flexible and powerful models that can incorporate symbolic knowledge without introducing domain-specific biases.

Acknowledgements

This work is supported by the project EVENFLOW – *Robust Learning and Reasoning for Complex Event Forecasting*, which has received funding from the European Unions Horizon research and innovation programme under grant agreement No 101070430.

References

[1] J. Agrawal, Y. Diao, D. Gyllstrom, and N. Immerman. Efficient pattern matching over event streams. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 147–160, 2008. ISBN 9781605581026. doi: 10.1145/1376616.1376634. URL https://doi.org/10.1145/1376616.1376634.

[2] E. Alevizos, A. Skarlatidis, A. Artikis, and G. Paliouras. Probabilistic complex event recognition: A survey. 50(5), 2017. doi: 10.1145/3117809.

[3] E. Alevizos, A. Artikis, and G. Paliouras. Complex event forecasting with prediction suffix trees. *VLDB J.*, 31(1):157–180, 2022. doi: 10.1007/S00778-021-00698-X.

[4] Apache. Flinkcep. URL https://nightlies.apache.org/flink/flink-docs-master/docs/libs/cep/. Version 2.2-SNAPSHOT, accessed July 2025.

[5] G. Apriceno, A. Passerini, and L. Serafini. A Neuro-Symbolic Approach to Structured Event Recognition. In *28th International Symposium on Temporal Representation and Reasoning (TIME 2021)*, volume 206 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 1–14, 2021. ISBN 978-3-95977-206-8. doi: 10.4230/LIPIcs.TIME.2021.11.

[6] G. Apriceno, A. Passerini, and L. Serafini. A Neuro-Symbolic Approach for Real-World Event Recognition from Weak Supervision. In *29th International Symposium on Temporal Representation and Reasoning (TIME 2022)*, volume 247 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 1–19, 2022. doi: 10.4230/LIPIcs.TIME.2022.12.

[7] G. Apriceno, L. Erculiani, and A. Passerini. A Neuro-Symbolic Approach for Non-Intrusive Load Monitoring. In *Volume 372: ECAI 2023*, Frontiers in Artificial Intelligence and Applications, pages 3175–3181, 2023. doi: 10.3233/FAIA230638.

[8] A. Artikis, M. J. Sergot, and G. Paliouras. An event calculus for event recognition. *IEEE Trans. Knowl. Data Eng.*, 27(4):895–908, 2015. doi: 10.1109/TKDE.2014.2356476.

[9] D. Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2012.

[10] M. Bucchi, A. Grez, A. Quintana, C. Riveros, and S. Vansummeren. CORE: a complex event recognition engine. *Proc. VLDB Endow.*, 15(9):1951–1964, 2022. doi: 10.14778/3538598.3538615.

[11] G. Cugola and A. Margara. TESLA: a formally defined event specification language. In *Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems*, pages 50–61, 2010. ISBN 9781605589275. doi: 10.1145/1827418.1827427.

[12] L. D'Antoni and M. Veanes. The power of symbolic automata and transducers. In *Computer Aided Verification*, pages 47–67, Cham, 2017. Springer International Publishing. ISBN 978-3-319-63387-9.

[13] A. Darwiche. Tractable boolean and arithmetic circuits. In P. Hitzler and M. K. Sarker, editors, *Neuro-Symbolic Artificial Intelligence: The State of the Art*, volume 342 of *Frontiers in Artificial Intelligence and Applications*, pages 146–172. IOS Press, 2021. doi: 10.3233/FAIA210350.

[14] L. Getoor and B. Taskar. *Introduction*. The MIT Press, 2007. ISBN 9780262256230. doi: 10.7551/mitpress/7432.003.0003.

[15] N. Giatrakos, E. Alevizos, A. Artikis, A. Deligiannakis, and M. Garofalakis. Complex event recognition in the big data era: a survey. *The VLDB Journal*, 29(1):313–352, 2020. doi: 10.1007/s00778-019-00557-w.

[16] A. Grez, C. Riveros, M. Ugarte, and S. Vansummeren. A formal framework for complex event recognition. 46(4), 2021. doi: 10.1145/3485463.

[17] D. Gyllstrom, E. Wu, H.-J. Chae, Y. Diao, P. Stahlberg, and G. Anderson. SASE: Complex event processing over streams. *CoRR*, abs/cs/0612128, 2006.

[18] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735.

[19] N. Katzouris and G. Paliouras. Answer set automata: A learnable pattern specification framework for complex event recognition. *ECAI*, 2023.

[20] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2017.

[21] E. Lokman, V. T. Goh, T. T. V. Yap, and H. Ng. Driving event recognition using machine learning and smartphones. *F1000Research*, 11, 2022. doi: 10.12688/f1000research.73134.2.

[22] W. Maddern, G. Pascoe, C. Linegar, and P. Newman. 1 year, 1000km: The oxford RobotCar dataset. *The International Journal of Robotics Research (IJRR)*, 36(1):3–15, 2017. doi: 10.1177/0278364916679498.

[23] N. Manginas, G. Paliouras, and L. D. Raedt. NeSyA: Neurosymbolic automata. to appeat at IJCAI, 2025. Pre-print doi: 10.48550/arXiv.2412.07331.

[24] R. Manhaeve, S. Dumančić, A. Kimmig, T. Demeester, and L. D. Raedt. Neural probabilistic logic programming in DeepProbLog. *Artificial Intelligence*, 298, 2021. doi: 10.1016/j.artint.2021.103504.

[25] D. Mitrović. Reliable method for driving events recognition. *IEEE Transactions on Intelligent Transportation Systems*, 6(2):198–205, 2005. doi: 10.1109/TITS.2005.848367.

[26] L. D. Raedt, K. Kersting, S. Natarajan, and D. Poole. *Statistical Relational Artificial Intelligence: Logic, Probability, and Computation*. 2016. ISBN 978-3-031-00446-9. doi: 10.1007/978-3-031-01574-8.

[27] J. Renkens, D. Shterionov, G. Broeck, J. Vlasselaer, D. Fierens, W. Meert, G. Janssens, and L. D. Raedt. ProbLog2: From probabilistic programming to statistical relational learning. In *Proceedings of the NIPS Probabilistic Programming Workshop,*, 2012.

[28] G. Singh, S. Saha, M. Sapienza, P. H. Torr, and F. Cuzzolin. Online real-time multiple spatiotemporal action localisation and prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3637–3646, 2017.

[29] G. Singh, S. Akrigg, M. Di Maio, V. Fontana, R. J. Alitappeh, S. Saha, K. Jeddisaravi, F. Yousefi, J. Culley, T. Nicholson, et al. ROAD: The ROad event awareness dataset for autonomous driving. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 2022. doi: 10.1109/TPAMI.2022.3150906.

[30] G. van Noord and D. Gerdemann. Finite state transducers with predicates and identities. *Grammars*, 4(3):263–286, 2001. doi: 10.1023/A:1012291501330.

[31] M. Vilamala, T. Xing, H. Taylor, L. Garcia, M. Srivastava, L. Kaplan, A. Preece, A. Kimmig, and F. Cerutti. Using deepproblog to perform complex event processing on an audio stream. 10 2021. doi: 10.48550/arXiv.2110.08090.

[32] B. W. Watson. Implementing and using finite automata toolkits. *Natural Language Engineering*, 2(4):295–302, 1996. doi: 10.1017/S135132499700154X.

[33] T. Winters, G. Marra, R. Manhaeve, and L. D. Raedt. DeepStochLog: Neural stochastic logic programming. *AAAI Conference on Artificial Intelligence*, 36(9):10090–10100, 2022. doi: 10.1609/aaai.v36i9.21248.

[34] T. Xing, L. Garcia, M. R. Vilamala, F. Cerutti, L. Kaplan, A. Preece, and M. Srivastava. Neuroplex: learning to detect complex events in sensor networks through knowledge injection. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*, pages 489–502, 2020. ISBN 9781450375900. doi: 10.1145/3384419.3431158.

[35] Z. Yang, A. Ishay, and J. Lee. NeurASP: Embracing neural networks into answer set programming. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 1755–1762. International Joint Conferences on Artificial Intelligence Organization, 2020. URL 10.24963/ijcai.2020/243.

[36] M. Z. Yazd, I. T. Sarteshnizi, A. Samimi, and M. Sarvi. A robust machine learning structure for driving events recognition using smartphone motion sensors. *Journal of Intelligent Transportation Systems*, 28(1):54–68, 2024. doi: 10.1080/15472450.2022.2101109.

[37] H. Zhang, Y. Diao, and N. Immerman. On complexity and optimization of expensive queries in complex event processing. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, pages 217–228, 2014. ISBN 9781450323765. doi: 10.1145/2588555.2593671.