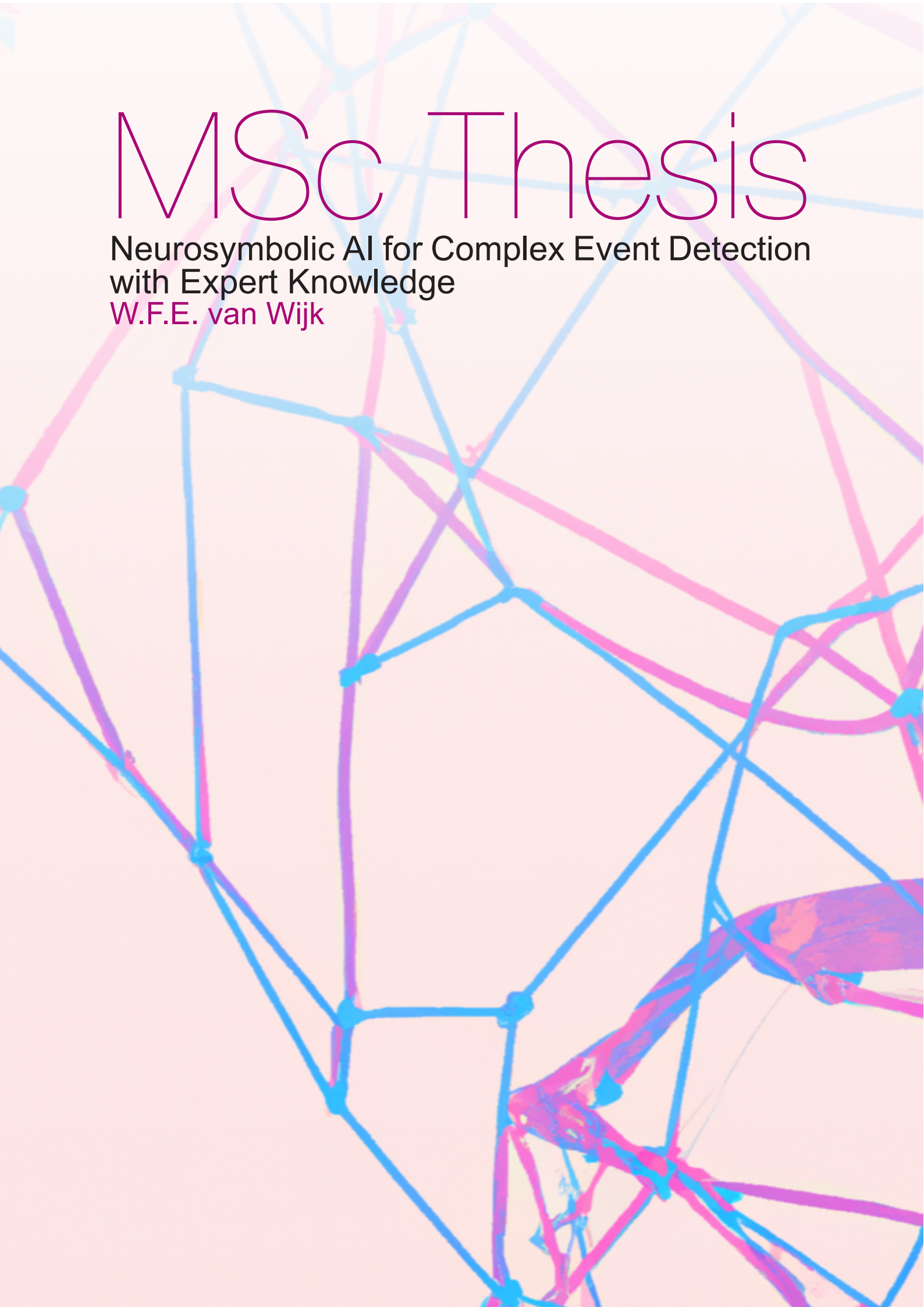
MSc Thesis

Neurosymbolic AI for Complex Event Detection with Expert Knowledge

W.F.E. van Wijk



MSc Thesis

Neurosymbolic AI for Complex Event Detection with Expert Knowledge

by

W.F.E. van Wijk

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday January 26, 2024 at 9:00 AM.

Student number:	4443403		
Thesis committee:	Chair & daily supervisor	Prof. dr. ir. J. Hellendoorn,	TU Delft
	Daily supervisor	Ir. D. Aljawaheri,	TNO
	Committee member	Dr. ir. C. Hernández Corbato	TU Delft
	Committee member	M. Boldrer	TU Delft

TU Delft

TNO

Contents

1 Introduction

Due to a rise in threats and organised attacks in the Netherlands, the pressure on the system 'Bewaken en Beveiligen' (translated to Surveillance & Security) has highly increased (Ministerie van Justitie en Veiligheid, 2022). This system is responsible for the protection of persons, buildings and services in the Netherlands against organised attacks or violation of the physical security. The combination of an increase in high profile persons that need safeguarding and the limited capacity of the workforce has had consequences for the quality of close protection of these persons (Onderzoeksraad voor Veiligheid, 2023). Therefore, there is a need to reinforce the system in general and more specifically the close protection of high profile persons. With regards to the system, the minister of Justice and Security wants a more flexible operational workforce that can adapt to the changes in threats such that the workforce is deployed at the right place at the right time. One way to achieve this is by investing in technologies that alleviate the workload of the workforce (Ministerie van Justitie en Veiligheid, 2022). To this end, the Royal Marechaussee of the Netherlands aims to automate residential security of high profile persons such that the workforce can be deployed more efficiently. The Marechaussee has a clear vision on what part of the process should be automated and what part should stay in the hands of the workforce. The situation will be as follows: the workforce will mainly be situated in a control room where they have real-time access to all sensors installed for surveillance. An intelligent surveillance model will autonomously detect suspicious events and alarm the workforce that can act accordingly. It is therefore important that the workforce has insight into why they have been alarmed and how the model came to that conclusion. Furthermore, the workforce needs to have trust in the system for a good cooperation. This can be achieved by mimicking the decision process of the workforce as much as possible such that errors are minimised (either false positives or false negatives) and only situations that are relevant are reported. If the system sends an alarm for every person that walks by and ties its shoe laces, the workforce will soon ignore the alarms. On the other side of the spectrum, if the system does not alarm enough, the workforce will end up doing all the work. Thus, it is crucial to have an accurate system that the workforce has trust in. Lastly, digital surveillance will ensure every movement around a residence is being recorded and if necessary saved. This will make it possible to reanalyse situations after they have happened. Surveillance can be made multi-modal, however, this research will solely focus on real-time video surveillance. By making it autonomous, we are speaking of Intelligent Video Surveillance (IVS).[1]

1.1. Application of the Research

With a rise in Closed-Circuit Television (CCTV) cameras comes a rise in IVS systems. These systems are used in many different areas such as crime prevention, shopping, airport monitoring, elderly care and traffic control (Shidik et al., 2019). Important with residential security is the relation between entities, not only in space but also in time. Alarming scenes can exist of a certain object or a certain action but will most likely be a combination of different actors and the spatio-temporal relation between them. For example: a person performing an action at a certain suspicious place, a person straying for a longer

[1] A considerate amount of information in this chapter has been taken from discussions with the Deep Vision team of the Marechaussee.

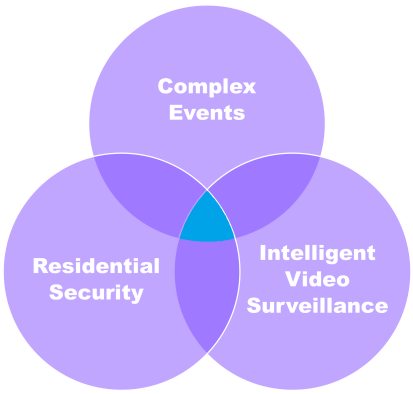


Figure 1.1: The blue area symbolises the application of this research.

time period around the residence, a group of people approaching the house or a sequence of normal actions which together are considered suspicious. The field of complex event processing looks into these spatio-temporal combination of simple events and how to detect them (Vilamala et al., 2023). We therefore focus on IVS for complex events. In conclusion, the application of this research will be on the intersection of IVS, residential security and complex events, see Figure 1.1, marked by the blue area.

1.2. Neurosymbolic Artificial Intelligence

To be able to recognise complex events from video data, we need a model that is able to process low-level data into meaningful abstractions such that it is human-understandable. This is where neurosymbolic Artificial Intelligence (AI) comes in. In short, neurosymbolic AI is a combination of subsymbolic deep learning components with symbolic knowledge reasoning. Combining these two types of AI ensures the best of both worlds. The deep learning part excels at recognising patterns and dealing with noisy data, while the symbolic system deals with high level, goal-oriented reasoning and extrapolation (Garcez and Lamb, 2023). There are multiple reasons why a neurosymbolic model is interesting for the use case of the Marechaussee:

1. The combination of neural networks with knowledge reasoning reduces the need for large amounts of training data. There is no need to learn a model from scratch, but instead new levels of abstraction can be reached with the reasoning part. Video surveillance is a data-scarce domain as there are not many accurately annotated libraries for infrequently occurring activities (Baxter et al., 2015). This makes a neurosymbolic model suited for the case of residential security.

2. Unlike the black box properties of deep learning, a neurosymbolic model has high explainability due to the use of human-understandable symbols and abstract reasoning. Furthermore, the model can exactly trace back how it arrived at a result by showing its line of reasoning. As stated before, this is a very important aspect for this case.

3. Lastly, a big advantage of using neurosymbolic reasoning is the ability to incorporate expert knowledge into the model in the form of logic. There are many experts on this case that would be happy to share their knowledge. This ensures not every bit of knowledge will have to be learned from the data.

However, the implementation of a neurosymbolic model for residential security comes with certain challenges. The first being how to exactly incorporate this knowledge such that the model is able to reason with it. There is a diverse collection of neurosymbolic models, types of logic and inference theories in which the choice of one influences the other. Additionally, there is a need within neurosymbolic research to be able to handle uncertainty (Sreelekha, 2018). First, the uncertainty that comes with sensory data such as noise or missing information (i.e. data uncertainty) and secondly the uncertainty that comes with the definition of events (i.e. pattern uncertainty) (Alevizos and Skarlatidis, 2017). The spatio-temporal relations of the different objects and actors that arise in surveillance footage can make

inference a difficult task. Add uncertainty to this mix and we arrive at a major challenge in the neurosymbolic AI field at this point, how to perform efficient probabilistic inference in neurosymbolic computation (Raedt and Kimmig, 2015).

1.3. Research Question

The combination of neurosymbolic AI and complex event detection is not new (Xing et al., 2020; Apriceno et al., 2022; Vilamala et al., 2023) and has seen an increase in popularity over recent years. An interesting direction in this research field is a type of neurosymbolic AI that maintains the full expressivity of the logic part while being end-to-end differentiable. Examples are Scallop (Li and Huang, 2023) or the neural probabilistic logic programming (nPLP) DeepProbLog (Manhaeve et al., 2018). These type of frameworks have been used for complex event detection but have not yet been tested on real-world video data (Vilamala et al., 2023). For this research, complex events will be detected with an nPLP framework due to their theoretical suitability. Furthermore, many researches on neurosymbolic AI for complex events assume a full video as input such that past, present and future knowledge can be used (Apriceno et al., 2021; A. Khan et al., 2019). In the case of residential security, we are limited to a system that works online, where data is processed as it comes in and knowledge about future events is not available. We then arrive at our research question:

"To what extent is a neural probabilistic logic programming framework suitable for online complex event detection on real-world surveillance video data while efficiently making use of available expert knowledge?"

This question will be answered in three steps:

Creating a proof-of-concept: Acknowledging that creating a model capable of performing well in all situations is an inestimable research, the first step focuses on delivering a proof-of-concept by using pre-trained models and pre-made software where possible and using a small subset of complex events. This ensures the research can be focused on the theoretical feasibility of using an nPLP framework instead of fine-tuning algorithms for higher detection rates.

Research on improving inference: Inference of probabilistic and temporal logic is known to be a costly task. We therefore look at ways to improve inference efficiency that conform to the limits set by the surveillance context.

Evaluation based on expressivity, efficiency and adaptability: The proof-of-concept as well as the theoretical knowledge gathered during the research will be evaluated based on expressivity, efficiency and adaptability.

DeepProbLog (Manhaeve et al., 2018) is a popular nPLP framework and has been used for complex event detection on synthetic or audio data. It is, however, limited in its ability to reason with temporal data. Therefore, we extend DeepProbLog with Prob-EC (Skarlatidis et al., 2015), a probabilistic extension of the Event Calculus (Kowalski and Sergot, 1985) used for temporal representations. We call this new framework the Hellenwaheri framework, named after the two driving forces behind this research. As intractability of the inference process is a common theme when dealing with probabilistic data and spatio-temporal relations, the gaps in improving inference for this framework will be identified. More specifically, the contributions of the paper are as follows:

1. Combining DeepProbLog with Prob-EC.

2. Designing an nPLP framework for online complex event detection on real-world data.

3. Presenting insights into improving DeepProbLog inference for online complex event detection on real-world surveillance video data.

1.4. Outline of the Thesis

We start this thesis by giving an introduction to IVS systems and neurosymbolic AI in Chapter 2 and how this connects to the case of complex event detection for residential security. Important features of an IVS system will shortly be treated as well as why neurosymbolic AI is a good fit for the case of IVS. In Chapter 3, the criteria that the neurosymbolic model should adhere to are discussed. Chapter 4 summarises the main findings of the literature research and how they connect to the current research. The background knowledge on Probabilistic Logic Programming (PLP) needed to understand the research is given in Chapter 5. In Chapter 6, the design of the framework is extensively described followed by the test set-up for the case of surveillance data in Chapter 7. Chapter 8 looks into different possibilities to improve the inference efficiency of the framework. The output of the framework and its processes are treated in Chapter 9. Lastly, the research will be discussed in Chapter 10 and concluded in Chapter 11.

2

Neurosymbolic AI for the case of IVS

The application of our research will be on the intersection of residential security, intelligent video surveillance (IVS) and complex events. A neurosymbolic framework will be used to detect the suspicious behaviour around the residences. These building blocks are introduced in this chapter.

2.1. Intelligent Video Surveillance

With an increase in the use of video surveillance technologies comes the need for more efficient video analysis. Since 2010, there has been an increase in research on surveillance systems that could automate (parts of) the video analysis pipeline. In the past, a human would be tasked to monitor all the cameras from a control room and act upon suspicious or abnormal activity. This is not only inefficient, but also proven ineffective and is constrained by the limits of human capability (Shidik et al., 2019). This is where IVS solutions come in. IVS is a method to automate the analysis of video data such that spaces can be monitored real-time with the goal of enhancing scene understanding. The system acts as a preliminary filter of potentially dangerous events (Persia et al., 2020) and will timely alert security personnel for validation. IVS is valuable for many applications and has already been used in domains such as crime prevention, elder treatment and accident detection (Shidik et al., 2019).

Not only is creating an IVS solution a challenging task on the hardware side (which we will not treat in this research), but also the different tasks on the software side that need to be performed on different levels make this a non-trivial issue. Many articles have been published that focus on the implementation of algorithms to perform singular tasks such as motion detection, object detection, object or event classification, object tracking, behaviour and activity recognition and database or system operations (Shidik et al., 2019). These researches have succeeded in alleviating the work of security personnel (for an overview, see Shidik et al., 2019). However, the complexity of the scenes and events happening in residential security requires an IVS system that is able to handle diverse types of information and therefore has multiple levels for processing information. There is also the issue that traditional surveillance systems trained in a specific context on specific events might be hard to generalise and are weak when it comes to detecting high-level events. Persia et al., 2020 states that a modern and effective video surveillance system has *"[...] to be flexible in capability to detect all the potential event occurrences happening in the monitored environment, and to be adaptable to the context features of the observed scene"*. This highlights an important challenge in building a robust IVS system.

2.1.1. Complex Events

Although anomaly detection would seem a good research direction for this case, the literature usually refers to anomaly detection as *"finding patterns in data that do not conform to expected behaviour"* (Nawaratne et al., 2020). In other words, information obtained from the raw data is compared to a normal situation and anomalies are detected based on the difference. The anomalies we are looking for will be provided by the workforce and incorporated in the model in the form of knowledge which means we will not be looking for patterns in raw data. As the two methods are fundamentally different, it seems irrelevant to compare the two and we will leave anomaly detection for what it is. A much more fitting research direction are complex events. Complex events are spatio-temporal combinations of multiple

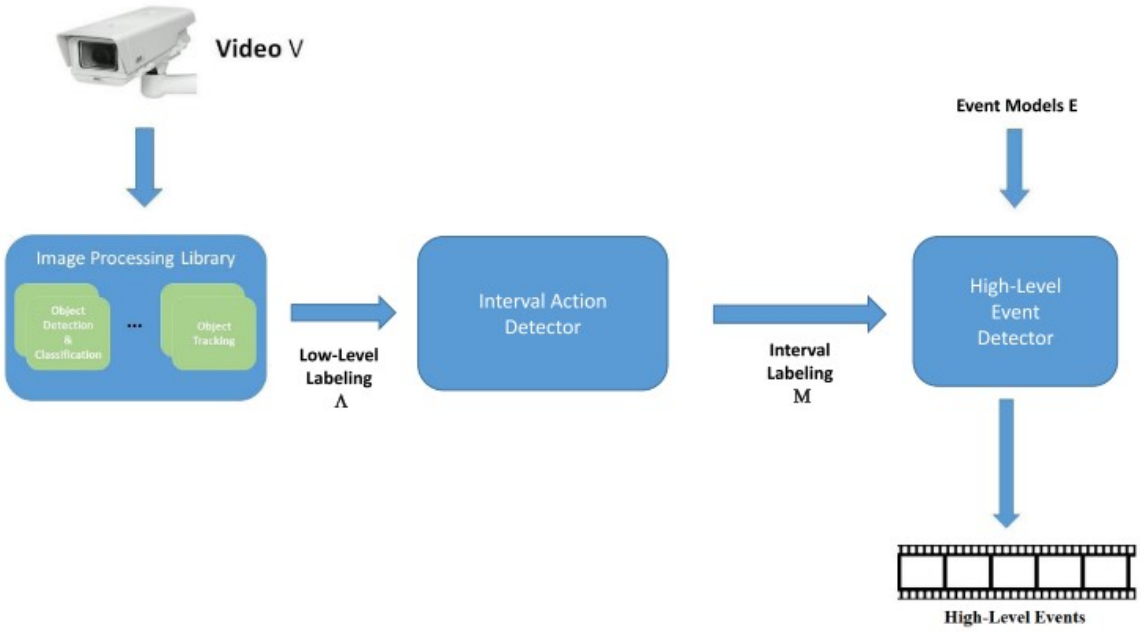


Figure 2.1: From Persia et al., 2017b, example architecture for an IVS system

simple events (Vilamala et al., 2023). For example, in video surveillance the events representing that some people are walking together at the same time (temporal relation), in close distance and in a similar direction (spatial relations), may indicate the situation that those particular persons are moving together (complex event).The term is not specific to computer vision tasks but is also used in, for example, detecting cyber attacks based on IP addresses (Alevizos and Skarlatidis, 2017). In the case of residential security, a simple event can represent a variety of concepts. It could be a location, a track, a behaviour, an action and so on. As suspicious behaviour around the residence will often not be a straightforward atomic event, models designed for complex events are of interest for this research.

Deep learning methods such as Long Short Term Memory (LSTM) or Convolutional 3 Dimensional (C3D) are popular for complex event detection. However, these methods tend to require larger amounts of data, especially when the complexity of the relations increases (Vilamala, 2022). Furthermore, Ahmad and Conci, 2019 state that the performance of deep learning frameworks for complex event detection in videos is generally lower than images due to the high variability of visual content over time. Deep learning models are limited when it comes to extrapolation or learning complex relations. This leads to the conclusion that deep learning alone will likely not solve the problem of automated residential security. There is a general consensus in the literature that to be able to detect complex events, models need to process low-level and high-level information (Kardas and Cicekli, 2017; Persia et al., 2017a; Ben Mabrouk and Zagrouba, 2018). Different terms are used to refer to these two levels: low- and high-level, raw data and symbols, atomic events and complex events. Furthermore, it is often not a two layer structure but a hierarchical composition extracting new types of information in every layer (Kardas and Cicekli, 2017; Persia et al., 2017a). For example, a first low-level layer can detect objects and persons and track these throughout frames. Then medium-level information such as speed, distance and direction can be extracted. From this, events such as run, walk and stand can be obtained. Finally, we can extract complex events such as meet, left object, fight. An example of such architecture is displayed in Figure 2.1 (taken from Persia et al., 2017b).

2.2. Neurosymbolic AI

For decades, there has been a clear separation between the worlds of data driven AI and knowledge driven AI where researchers in both groups believed their methods would be the best form of AI. Data driven AI is also referred to as the connectionist approach and knowledge driven AI as the symbolic approach. Figure 2.2 shows how an apple could be interpreted for both approaches. It is no secret

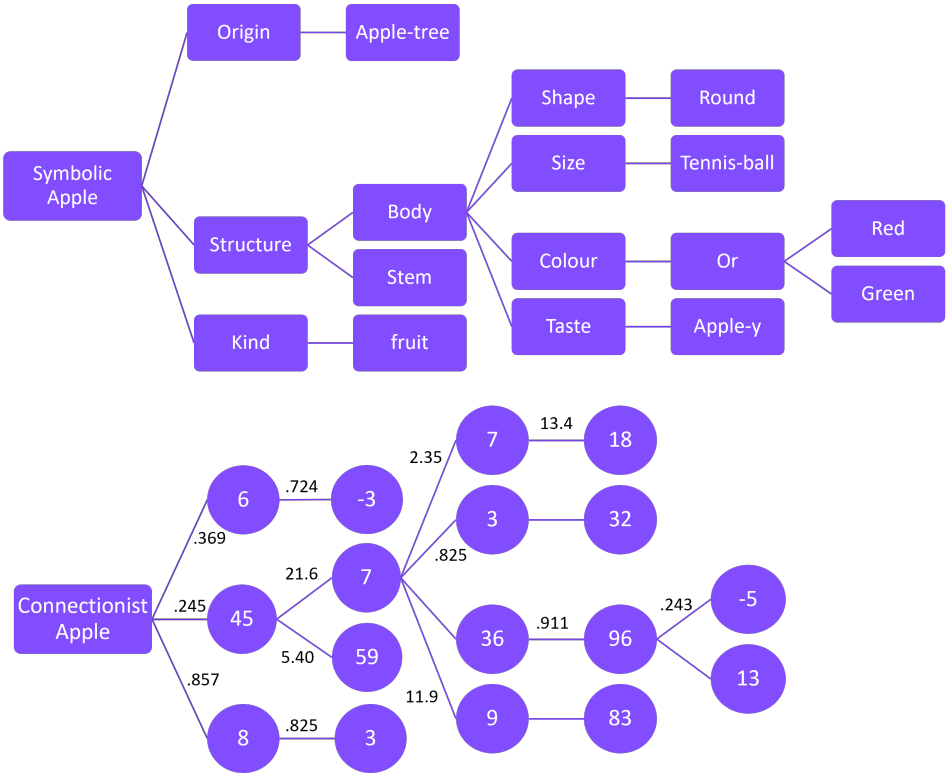


Figure 2.2: Representation of an apple in the symbolic and connectionist paradigm.

that deep learning, as part of the data driven approach, has achieved groundbreaking results in applications such as computer vision, game playing and natural language processing (Hitzler et al., 2023). However, as we have seen in section 2.1, deep learning is not suited for all tasks and researchers are agreeing that combining the world of data driven AI and knowledge driven AI might bring us closer to a higher level of intelligence. Interestingly, both methods have complementary strengths and weaknesses.

Deep learning systems are distributed and continuous and their powers lie in their ability to find patterns in noisy data. Due to their learning capabilities, neural networks have been able to solve many challenges we face today. It is therefore not surprising that the interest in machine learning from both industry and scientific parties has grown enormously. There is, however, a reoccurring theme in debates about this type of AI: their lack of explainability. Especially now when the general public is becoming more and more concerned about the power of AI, transparent AI systems are essential. Furthermore, most neural networks need incredible amounts of data to be trained and function properly. This lack of parsimony requires an extensive amount of computational power and unacceptable levels of energy consumption [1](Garcez and Lamb, 2023). Lastly, deep learning models are weak when generalising beyond the training distribution. In conclusion, deep learning techniques have been proven data hungry, shallow, and limited in their ability to generalise (Marcus, 2020).

This is where symbolic reasoning comes into play. Symbolic AI is localist and discrete. Using symbolic knowledge offers the possibility to integrate expert knowledge into the program. Furthermore, explainability is improved as the reasoning steps can be retraced and the model can be interpreted or validated. Extrapolation of knowledge is also a feature which favours knowledge reasoning (Garcez and Lamb, 2023). However, knowledge reasoning as we know it, is slow compared to machine learning, cannot handle noisy data well and is limited in size due to the need of explicit knowledge-bases (Hitzler et al., 2023).

With knowledge driven AI bringing the first wave of AI in the 1980s, data driven AI bringing the second wave in the 2010s, we are now in the third wave that fuses both worlds and brings neurosymbolic

[1]https://www.theguardian.com/commentisfree/2019/nov/ 16/can-planet-afford-exorbitant-power-demands-of-machine- learning

AI (Hitzler et al., 2023). In the past two decades, the field of neurosymbolic AI (sometimes also referred to as neural-symbolic AI) has gained increasing attention among researchers. Combining learning and reasoning, low-level and high-level knowledge, data driven and knowledge driven computing, statistics and logic, theory and data, subsymbolic and symbolic paradigms results in a form of AI which is more powerful than the parts of its sum. Learning can be used for efficient perception and pattern recognition whereas reasoning can be used for explainability, extrapolation and planning (Hitzler et al., 2023). For an overview of current applications such as healthcare, finance, natural language processing, image processing and corresponding literature, see Bouneffouf and Aggarwal, 2022. Neurosymbolic AI has proven to converge quicker needing less training data for the same or better accuracy compared to deep learning (Mao et al., 2019; Vilamala et al., 2020). There are, however, also challenges when integrating the connectionist and symbolic approaches mainly being: effective integration, expressive reasoning and robust learning (Hitzler et al., 2023). Additionally, Susskind et al., 2021 states that neurosymbolic models have less potential for parallelism which can cause longer training times.

As already touched upon in the introduction, neurosymbolic AI possesses multiple characteristics that makes it very suitable for our application:

Explainability As AI is becoming a hot-topic, so is the term explainability. One of the reasons the general public fears AI is due to its black box characteristics. It often remains a challenge to know how the model has arrived on a particular result. This feature is of even greater importance when making decisions about a type of threat that could have severe consequences as is the case in residential security for high profile persons. The workforce needs to know why the model has chosen to send out an alarm and what exactly was the contribution of each and every component. Once the alarm has been triggered, decisions need to be made in split seconds. Neurosymbolic models give the opportunity to translate the deep learning outcomes to symbols that are interpretable by the workforce.

Data efficiency Unfortunately (or luckily), there is not a great amount of data available on the subject of anomalous behaviour. Training a model for a variety of context, temporal relations and complex relational structures would require enormous amounts of data with a more traditional deep learning approach. According to the literature, neurosymbolic models need less data to be trained (Mao et al., 2019; Vilamala et al., 2020) as reasoning permits abstraction and extrapolation.

Incorporation of knowledge Lastly, it would be a waist to not incorporate the amount of knowledge readily available from the workforce. With years of experience, the workforce has gathered knowledge that will save training time and provide some guidance by incorporating meaningful semantics. As we are trying to mimic the decision process of the workforce, their expert knowledge is enormously valuable.

In summary, both the field of neurosymbolic AI as well as the field of IVS have seen an increase in popularity due to the challenges of today's world. The IVS system needs to be able to detect complex events and deal with different levels of data. Neurosymbolic AI is exactly able to do that by reasoning with both low- and high-level data. Designing a robust yet expressive neurosymbolic framework for the case of IVS remains a challenging task.

3

Criteria of the model

By combining the information from the literature and the Marechaussee and integrating these with the application of residential security, we arrive at six criteria that the neurosymbolic model should adhere to. Each criteria will be elaborated on in this chapter and together form the backbone of the research. The criteria are discussed in no particular order and are all essential for a good functioning of the IVS framework.

3.1. Integrate Expert Knowledge

There are several advantages that come with Neurosymbolic AI and more specifically with the use of symbols. One of them is the ability to understand the model and the decisions it makes as symbols are human understandable. A second is the possibility to integrate expert knowledge. This reduces the complexity of the learning process and therefore requires less training data (Vilamala, 2022). In many industries, the need for expert knowledge can be a hurdle due to a lack of expertise. However, for the specific application of residential security for the Marechaussee where there is an entire workforce ready to implement their knowledge, the ability to integrate expert knowledge is a big advantage. This brings along some challenges such as how to properly translate this knowledge and ensure the detections remain unbiased.

3.2. Probabilistic

Uncertainty is an unavoidable aspect when dealing with real-world complex events and arises at multiple levels in the system. First, there is the uncertainty around noisy observations or due to incomplete inputs such as after a sensor malfunctioning. This type of uncertainty is categorised as data uncertainty. Secondly, there can be uncertainty in the definition of the complex event, termed pattern uncertainty (Alevizos and Skarlatidis, 2017). For example, when a simple event as part of a complex event definition is not detected, or not detected at the right moment. In this research we will mainly focus on data uncertainty and recommend dealing with pattern uncertainty as future research. For an overview of different symbolic event recognition methods for all types of uncertainty, see Skarlatidis and Vouros, 2014.

One approach to deal with noisy low-level input is to determine a confidence threshold that detections or simple events must exceed to be accepted by the model, facilitating the use of a deterministic model. The disadvantage of this method is that the information contained in the confidence score is not exploited by the high-level system. This in turn leads to suboptimal event recognition (Skarlatidis and Vouros, 2014). Processed low-level data usually has a confidence estimate attached to it (Mantenoglou et al., 2023). Following Raedt et al., 2023, this confidence score can be interpreted as a probability distribution. These probabilities can then be translated (using inference) to higher levels and are accounted for in the output. Probabilistic methods are known to provide a rigorous and unified framework for processing uncertainty (Alevizos and Skarlatidis, 2017). Pattern uncertainty is handled partially by using a probabilistic system instead of a crisp system. Simple events are therefore not true or false, but are true with a certain probability which is being forwarded in the inference process.

3.3. Online Processing

Research on complex event recognition can be divided along two themes: online processing or offline processing of input data. Offline processing is common in, for example, analysing sports matches. The analysis happens when the full video is available and one of the main tasks is knowing in what time frame the complex event happened (Apriceno et al., 2022). In contrary, online recognition happens while the stream is being processed. Examples can be found in surveillance or autonomous driving (Suchan et al., 2021). This is what happens in the case of IVS. The workforce needs to be alarmed almost immediately while a complex event is starting. This brings the challenge of not having information available about future time steps, thus there is no point at which we can say with certainty that we have all the information (Vilamala, 2022). Preferably, the system should work real-time such that it is able to keep up with the incoming stream, however, this will not be a priority for now.

3.4. Reasoning with Spatio-Temporal Data

The use of video data offers the possibility to analyse temporal events that contain more information than the still images and can therefore provide more insight into understanding scenes. Furthermore, spatio-temporal formalisms are naturally grounded in physics and human cognition and should be a minimal requirement for an intelligent system to be able to generalise in a causally sound manner (Hitzler et al., 2023). To make optimal use of this, we need a model that is able to deal with temporal information and knowledge. Time representation can be made explicit or implicit, meaning respectively time itself is considered a variable or not (Alevizos and Skarlatidis, 2017). Our focus is on explicit representations as this provides important information for the workforce. The spatial aspect mainly refers to being able to detect multiple entities and their relations in one frame. This gives the possibility to capture interactions between persons and objects and use features such as tracking and re-identification. Additionally, we have the possibility to detect multiple simple events per frame as opposed to limiting the detection to one event per frame (Preece et al., 2021).

3.5. Focus on Logic

There is also a common subdivision in the literature of neurosymbolic models that focus on the neural aspects or neurosymbolic models that focus on the logic aspect. The former consists of architectures in which the logic is compiled into the neural network (LTN, TensorLog) or architectures that turn the logic into a regularisation function and act as constraints (SL, SBR). When logic is used as a constraint, it acts as a soft constraint which means there is no guarantee it will be followed. The latter consists of logic programming frameworks extended with neural networks to allow differentiable operations (DeepProbLog) or frameworks that are turned into differentiable versions (∂ILP, NTP) (Raedt et al., 2020). Raedt et al., 2023 states that to retain full expressivity of both the neural part as the reasoning part, the two parts should not be integrated. Similarly, Latapie et al., 2022 states that the transformation of symbolic representations into subsymbolic models brings an inevitable loss of the underlying causal model. Therefore, we favour logic programming frameworks extended with neural networks where the logic retains full expressivity. As the raw input data from the videos is in continuous space, the neural networks will take care of the first processing step. On the other hand, the discrete nature of the symbolic system consisting of logical descriptions is more suited for the reasoning step.

3.6. End-to-end Differentiable

Lastly, the model should be end-to-end differentiable allowing simplified back propagation through the model. Making the full pipeline differentiable removes the need for supervision on every level (Apriceno et al., 2021). The full model can then be trained with only supervision on the output (the complex event probabilities). Obtaining labels for every simple event is a tedious task compared to obtaining labels for the complex events. First, because there is a much larger number of simple events in the same input data and secondly, labelling the occurrence of a simple event is not as straightforward as labelling when a complex event occurs due to larger differences in how the event is interpreted (Vilamala, 2022). Furthermore, low-level labelled data is uncommon in the surveillance industry. This criteria is not necessary for a proper functioning of the system, however, it greatly simplifies training and therefore makes the framework more usable.

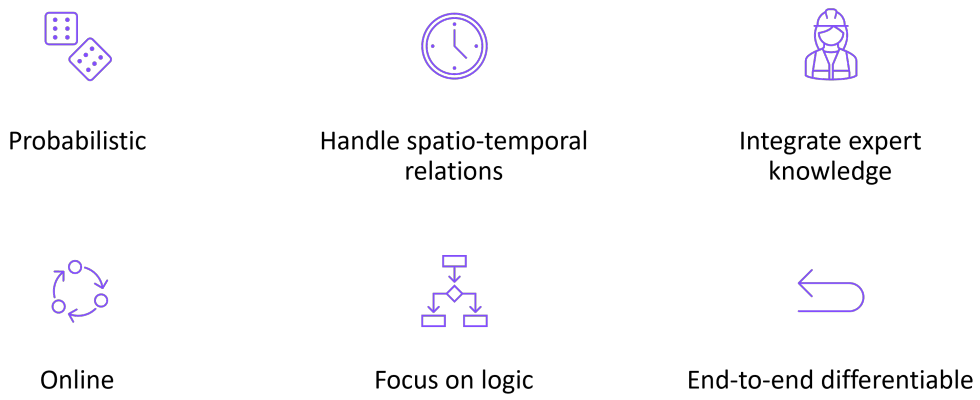


Figure 3.1: The six criteria the chosen framework should adhere to.

The six criteria are displayed in Figure 3.1. As these criteria will influence the framework chosen, they play a significant role in this research. Most of them will be discussed in different chapters in this research.

4

Literature Review

We have approached the literature on neurosymbolic AI for complex events from three angles: a theoretical, a practical and a detailed analysis. All three will be treated in this chapter as well as their conclusions. The theoretical part analyses the literature from, you guessed it, a theoretical perspective. Here, we will not focus on practical examples but on general knowledge that has been gathered over the years. In the practical part, we will look at researches that resemble ours and briefly describe their methods and conclusions. The last part, the detailed analysis, evaluates five frameworks based on the criteria discussed in Chapter 3. We end this chapter with a summary and how this shapes the research.

4.1. Theoretical Analysis

A domain that has been concerned with integrating logic with probabilistic reasoning is Statistical Relational AI (StarAI). StarAI looks at methods that can represent, reason and learn with both uncertainty and complex relational structures (Skarlatidis and Vouros, 2014). Not surprisingly, Marra et al., 2021 have written a survey with a comparison between neurosymbolic AI and StarAI, focused on a logical perspective. Methods are compared based on seven dimensions. We will discuss two of these dimensions as they are most relevant to our research.

4.1.1. Logic vs. Probabilistic Paradigm

The first dimension is about what paradigm (either probabilistic or logic) is more preserved. By preserving is meant to what extent the model of the original paradigm can be replicated. The survey states two approaches for integrating probabilities with logic within the domain of StarAI. The first is based on probabilistic graphical models (PGM) where the probabilistic paradigm is fully preserved and logic is added only to generate the PGM. An example is Markov Logic Networks (MLN). Technically, they do not use probabilities but weights for the rules. This makes them less intuitive and often the weights have to be learned. Furthermore, the flexibility of these type of models can also increase their computational complexity due to the possible combinations of variables and time points. However, different methods have been developed that deal with this complexity.
The second is based on probabilistic logic programming (PLP) where the logic paradigm is fully preserved and probabilistic inference is added. PLPs are built on top of Prolog and combine the expressive power of programming languages and graphical models. The most well-known PLP language is ProbLog (De Raedt et al., 2007), a probabilistic extension of Prolog. These StarAI methods employ a variant of Sato's distribution semantics and are therefore able to deal with uncertainty (Raedt and Kimmig, 2015). This makes the inference more complex but maintains the power of the logic.

4.1.2. Model-Theoretic vs. Proof-Theoretic

The second dimension treats model-theoretic vs proof-theoretic inference. Model-theoretic inference first grounds the logic by replacing the variables in the clauses with constants. The key idea is to find truth assignments (or models) of symbols such that they satisfy a given theory. Model-theoretic techniques use logic as a constraint and can be mapped to undirected graphs, or Markov Networks.

This compares to neurosymbolic methods that use full clausal logic. Proof-theoretic inference has the goal to provide proofs for a certain query by performing a sequence of inference steps and uses the probability distribution. Proof-theoretic techniques use logic for inference and can be mapped to directed models, or Bayesian networks. This compares to neurosymbolic methods that use definite clauses and causal logic. Models such as PLP and ∂ILP are proof-theoretic while extended versions of MLN are model-theoretic.

Grounding, as in model-theoretic reasoning, is claimed to either result in exponential explosion of the number of boolean variables or severe limitations to the expressive power of the logic (Hitzler et al., 2023). However, model-based approaches can often handle more scalable systems at inference time (Marra et al., 2021). With proof-based systems, there is full control and understanding of how the model uses the logic as it acts as structural constraint and is not encoded in the weights of the network as with model-based approaches. Therefore, logic can be extended or modified at test-time without the need to retrain (Marra et al., 2021). These systems also use backward chaining as logical inference. In contrast to forward reasoning that reasons for all possible worlds, backward reasoning makes use of the constraints imposed by a query and only considers those relevant worlds. This can make backward reasoning more efficient, especially in the case of uncertainty.

4.2. Practical Analysis

The world of complex event recognition and neurosymbolic AI is complex and involves many different frameworks. Each with their own representation, reasoning, inference and learning methods. This section is dedicated to discussing some researches that use logic for complex event detection.

An example of a framework that uses a very intuitive way of detecting complex events in a surveillance domain is presented in (Persia et al., 2017a). The model is able to efficiently handle time series and interval data by using temporal relational algebra to process events. The raw data is transformed to medium-level events using image processing techniques and the high-level event detector combines the medium-level annotations with the event knowledge base to detect whether one of the known complex events happen in the labelling. Furthermore, there is a GUI on every level in the model such that the user can interact with the system. In Caruccio et al., 2019, a similar approach is used but it also provides the modelling of context. The integration of context means different scenarios can be interpreted based on their context. The framework does deal with uncertainty and spatio-temporal relations to some extent but not as much as we would like. For instance, for the temporal relations, entry conditions and effects are used but these are not expressive enough. For uncertainty, they handle pattern uncertainty by differentiating between mandatory or optional simple events. Data uncertainty is handled as a special property in the description of an element of context or a general context descriptor. In J. Khan et al., 2020, a neurosymbolic framework is used for event processing. Visual and commonsense reasoning is used to expand on a (temporal) knowledge graph for a complete scene representation, after which it is matched to the queried high-level events. A simple approach can be found in A. Khan et al., 2019 where the Event Calculus is used in combination with Answer Set Programming (ASP) to detect certain events in video fragments of soccer matches. They conclude that the performance of their approach is mainly dependent on how strict the complex events are defined. Although working with real-world video data, the framework does not handle uncertainty in any way. These four solutions present good results for detecting complex events, however, an important aspect of our model is that it should be able to handle uncertainty which all three models do not consider.

In Skarlatidis et al., 2015, Prob-EC is used to detect complex events based on rules defined with the programming language ProbLog. It takes labelled simple events as input and a set of rules defined by a domain expert. There are two downsides to this method. The first being that all simple events need to labelled in the input videos, the second being that there is no learning mechanism.

The approach used in Apriceno et al., 2022 is very similar to our case in many ways. Instead of using DeepProbLog, the authors create their own neurosymbolic model with temporal and probabilistic relations. The knowledge is represented by a variation of the Event Calculus and determines the definition of several complex events as well as soft and hard constraints on the sequence and duration of events. The neural part gives a probability for every complex event in the input video and a probability for every simple event happening in every time series. Based on these probabilities, the complex and (the timing of) the simple events are determined by solving a Mixed Integer Linear Programming (MILP) problem. The neural network is trained to recognise simple and complex events by minimising a loss

function and using gradient descent to update its weights. However, it lacks spatial relations with multiple actors and assumes only one simple event per time frame. Furthermore, the event recognition happens offline.

A military setting for a neurosymbolic model is presented in Preece et al., 2021, where DeepProbCEP, based on DeepProbLog, is used to detect complex events in audio and video files. The method seems promising with great results, however, the test cases are greatly simplified. Once again, it lacks spatial relations with multiple actors and assumes only one simple event per time frame.

Önal et al., 2013 propose a MLN framework which can handle probabilistic and temporal data to reason about complex events in surveillance. It also allows dynamic state changes. The downside of MLN, however, is that it scales badly with the number of predicates and is not very well suited for temporal relations as it considers time variables in the same way as other variables (Kardas and Cicekli, 2017). This is solved in Kardas and Cicekli, 2017 with approximate inference. With IVS as application, this research has interesting features. The framework has many properties we are looking for such as a hybrid model, hierarchical abstraction layers and uncertainty handling. However by using many separate building blocks, the model is not end-to-end differentiable.

Hitzler et al., 2023 proposes to annotate every spatio-temporal variable with the probabilistic values from the low-level processing and implement a reasoner that will support this. It is described as a promising yet challenging task. As a more simple alternative, the authors suggest to look into neurosymbolic probabilistic frameworks that can encode spatio-temporal formalisms. Examples are DeepProbLog and NeurASP.

4.3. Detailed Analysis

To identify the scientific gap related to our research, a more detailed analysis of comparable cases is necessary. Five frameworks have been selected due to their popularity or due to their resemblance in some part to our goals. They all have a reasoning part and are applied to either complex event detection or surveillance applications. The frameworks are evaluated based on the criteria discussed in Chapter 3. First, a short description per framework is given after which the conclusions of the detailed analysis will be presented. Table 4.1 shows how the different frameworks score on the criteria.

1. **Kardas and Cicekli, 2017: SVAS: Surveillance Video Analysis System** - This paper introduces a Surveillance Video Analysis System (SVAS) that is able to learn rules and complex event definition from complex video events. The Interval-Based Spatio-Temporal Model (IBSTM) is presented as an event modeling framework, bridging the semantic gap between humans and video computer systems. SVAS eliminates the need for predefined thresholds in scene or event models, distinguishing it from many existing studies. The proposed approach employs a hybrid set of machine learning techniques, including Threshold Models for features, Bayesian Networks, Bag of Actions, Highly Cohesive Intervals, and Markov Logic Networks.

2. **Apriceno et al., 2022: A Neurosymbolic Approach for Real-World Event Recognition from Weak Supervision** - This paper presents a neurosymbolic approach that exploits knowledge about the events and their temporal relations at training and inference time. A neural network predicts both if a complex event is happening as well as what simple events are happening in which time frame. The framework is probabilistic and does not only look at the order or duration of simple events (hard constraints) but also at obtaining the minimal cost (soft constraints). It uses a first guess for the complex event happening to look for interpretations of that complex event in time.

3. **Mantenoglou et al., 2023: Online event recognition over noisy data streams** - This paper proposes a framework that computes the most likely max intervals (PMI) for a complex event. It uses Prob-EC for initial processing after which PIEC is used for finding the intervals. They designed a special version of PIEC (oPIECbd) that works online (o) with a bounded support set (b) and leverages interval duration statistics to resolve memory conflicts (d). It proves that using intervals, as opposed to point-based systems, are much more efficient.

4. **Xing et al., 2020: Neuroplex: Learning to Detect Complex Events in Sensor Networks through Knowledge Injection** - This paper introduces a neurosymbolic framework that splits learning into a perception phase and a reasoning phase. For the perception task, it trains deep

neural networks to acquire low-level symbolic concepts while allowing the incorporation of symbolic human knowledge for high-level reasoning. The entire model can be trained end-to-end. Symbolic knowledge is expressed through finite state machines and logical rules.

5. **Caruccio et al., 2019: EDCAR: A knowledge representation framework to enhance automatic video surveillance** - This paper introduces the Elements and Descriptors of Context and Action Representations (EDCAR) knowledge representation framework. It enables the representation of a context and the potential events that might occur in it and works online. Complex events are modelled by a composition and sequence of actions with different occurrence types (mandatory, optional, one of, repeatable) and context.

First author	Publication year	Neurosymbolic /Hybrid	Probabilistic	Temporal	Spatial	Focus on logic	Online	Integrate expert knowledge	End-to-end differentiable
Kardas	2017	+	+	+	+	+	+	+	-
Apriceno	2022	+	+	+	-	-	-	+	+
Mantenoglou	2023	-	+	+	+/-	+	+	+	-
Xing	2020	+	+/-	+	+/-	-	+	+	+
Caruccio	2019	-	+/-	+/-	+	+	+	+	-

Table 4.1: Comparative analysis based on the criteria discussed in Chapter 3 for similar researches

A first thing we notice is that, although in different ways, all frameworks allow the integration of expert knowledge. This does not come as a surprise since the frameworks have been chosen due to their reasoning part. Secondly, the neurosymbolic models, namely 2 and 4, do not focus on logic. The logic is either copied by a neural network (4) or acts as a constraint on the loss function (2). Furthermore, when the framework does maintain the full expressivity of the reasoning part, the framework is not end-to-end differentiable. The list of criteria contained both end-to-end differentiable and focus on logic.

4.4. Scientific Gap

From the theory it becomes clear that a model theoretic approach that retains the logic paradigm is favourable for the case of IVS. These are brought together in the field of probabilistic logic programming (PLP). Even though a certain type of model is better suited for an inference task, exact inference for temporal relations and uncertainty can still become intractable (Marra et al., 2021; Kastrati and Biba, 2019; Kardas and Cicekli, 2017; Raedt and Kimmig, 2015). Up to date, this remains an open challenge for both directed and undirected graphical models. A suitable method should be expressive while keeping computational complexity to a minimum. From the comparable cases, we conclude there are many different ways to tackle the problem of neurosymbolic AI for complex events. Each having their own advantages, disadvantages and assumptions. Where one framework falls short on inference speed, another falls short on the expressiveness of logic or the ability of the model to learn. Lastly, from the detailed analysis we conclude none of the analysed frameworks combines all criteria. Especially, the combination of focusing on logic while being end-to-end differentiable is rare. Frameworks found in the literature that do combine these two aspects are neurosymbolic frameworks such as Scallop (Li and Huang, 2023), DeepProbLog (Manhaeve et al., 2018), DeepStochLog (Winters et al., 2022) and DeepProbCEP (Vilamala et al., 2023). While DeepProbLog and DeepProbCEP are based on Prolog (both are PLP frameworks), DeepStochLog is based on stochastic grammars and Scallop is based on Datalog and the theory of provenance semiring.

Summing up the conclusions from this chapter leads to the idea of using a neurosymbolic PLP-based framework for the detection of complex events. These types of frameworks are theoretically suited for scenarios where probability and complex spatio-temporal relations are present. Furthermore, they adhere to all criteria presented in Chapter 3. DeepProbLog is a well-known and well-maintained neural

probabilistic logic programming (nPLP) framework. Using DeepProbLog for complex event detection has already been done in Apriceno et al., 2021 and Vilamala et al., 2023. Both researches prove the ability of DeepProbLog to handle uncertainty and temporal relations and integrate expert knowledge. However they lack expressivity in how spatio-temporal relations are defined, do not work online and have only been tested on synthetic or audio data. In DeepProbCEP (Vilamala et al., 2023) for example, where a sequence of two MNIST digits is detected in a stream, the temporal relation is only defined by the order of digits and only one simple event per time point is allowed. For these reasons, DeepProbLog will be used in this research to perform online complex event detection on real-world surveillance data. To improve expressivity on temporal relations, DeepProbLog is combined with Probabilistic Event Calculus (Prob-EC) (Skarlatidis et al., 2015). To improve expressivity on spatial relations, we allow the detection of multiple events per frame and use a hierarchical system in the complex event definitions. We then arrive at the Hellenwaheri framework: a neurosymbolic framework capable of detecting complex events online from real-world video data. In the next chapter, we will go into depth on the working of DeepProbLog and Prob-EC.

5

Probabilistic Logic Programming

Both DeepProbLog and Prob-EC are extensions of the PLP language ProbLog. This chapter is dedicated to introducing PLP, ProbLog, Prob-EC and DeepProbLog as well as their semantics and inference process.

5.1. Introduction to PLP

Probabilistic Logic Programming (PLP) is a powerful paradigm that combines elements of logic programming and probability theory to model and reason about uncertainty. It combines the expressive power from directed graphical models and programming languages (Fierens et al., 2013). PLP has been studied for over 30 years and knows many different frameworks and languages (see Raedt and Kimmig, 2015 for an overview). Typical PLP languages use a form of Sato's distribution semantics (Taisuke, 1995) which is a generalisation of the least model semantics. Under the distribution semantics, a logic program defines a probability distribution over a set, termed a world. We can then calculate the probability of an atom A for a finite number of worlds by identifying the proportion of the worlds whose model contains A as true (Riguzzi and Swift, 2018). The distribution semantics are expressive enough to represent Bayesian networks, Markov chains and hidden Markov models (Raedt and Kimmig, 2015).

5.2. ProbLog

ProbLog (Kimmig et al., 2011) is a probabilistic extension of Prolog and is regarded as a very expressive directed graphical modelling language. Being Turing-complete, ProbLog is agile and provides flexibility in how the rules can be defined (Vilamala, 2022). The distribution semantics is well-defined for infinitely many ground probabilistic facts, however, we focus on the finite case. ProbLog uses negation as failure which means the negation of an atom is true exactly if the atom cannot be derived from the program. The Prolog convention is followed by starting variables with an upper case and the rest with a lower case. (Manhaeve, Dumančić, et al., 2021)

A ProbLog program consists of probabilistic facts of the form $p_i :: f_i$ and definite clauses, or rules. The probability of a ground instance of the fact f_i being true is p_i and false is $1-p_i$. These facts represent random variables and are assumed mutually independent. This means that a definite clause, which is defined as a conjunction of these random variables, has a probability equal to the the product of the probabilities of these random variables. A ground logic program L (meaning all variables have been replaced) then has a probability distribution equal to:

$$P(L) = \prod_{f_i \in L} p_i \cdot \prod_{f_i \notin L} (1 - p_i) \tag{5.1}$$

We call this subset L a possible world. As each possible world has a unique least Herbrand model, $P(L)$ can be used to define the success probability of a query. That is $P_s(q)$ is equal to the sum of the probabilities of the grounded programs that entail it (Vilamala, 2022):

Predicate	Meaning
happensAt(E,T)	Event E occurs at time T
holdsAt$(F=V,T)$	The value of fluent F is V at time T
initiatedAt$(F=V,T)$	At time T, a period of time for which $F=V$ is initiated
terminatedAt$(F=V,T)$	At time T, a period of time for which $F=V$ is terminated

Table 5.1: Main predicates of the Event Calculus.

$$P_s(q) = \sum_{L \models q} P(L) \tag{5.2}$$

This formula can be transformed to an equivalent propositional logic formula on which weighted model counting (WMC) can be performed. WMC refers to the process of calculating the sum of the weights that correspond to models that satisfy this propositional logic formula. Performing inference on this formula directly is not efficient (Manhaeve, Dumančić, et al., 2021; Skarlatidis et al., 2015). This formula cannot simply be transformed to a sum of products. This would entail that all different proofs are disjoint and thus represent mutually exclusive possible worlds which is not true in the general case (Skarlatidis et al., 2015). This is referred to as the disjoint-sum problem and is known to be #P-hard. ProbLog uses knowledge compilation techniques to achieve scalable inference (Vilamala, 2022). Different knowledge compilation methods have been used over the years, each time improving the inference speed. Most recently Sentential Decision Diagrams (SDD), a graphical representation of a propositional logic formula, are used to perform efficient inference in polytime.

Another feature supported by ProbLog is the use of non-ground probabilistic facts and annotated disjunctions (ADs). ADs are expressions of the form $p_1 :: h_1; ...; p_n :: h_n : -b_1, ..., b_m.$ with $\sum_i p_i \leq 1$, where p_i are the probabilities of the corresponding atoms h_i and b_j are literals. If all b_i are true, one of the heads h_j is true according to the probability p_j or none of them is true with probability $1 - \sum_i p_i$. ADs do not change the expressivity of ProbLog as they can be transformed into a set of independent facts and logical rules. Alternatively, ADs can be used to specify conditional probabilities. (Manhaeve, Dumančić, et al., 2021)

There are basically three tasks that can be performed with ProbLog (Fierens et al., 2013). The first is computing the marginal probability of a query. The model is given a query and we want to know what the probability is of the query being true over all possible worlds. The second is conditional probability in which we want to the probability of a query being true given some evidence. The last is MPE (most probable explanation) inference in which we want to find the world for which the probability of a query is highest, so the most probable world. In this research, ProbLog will be used for the first task. In Chapter 6, the second task will shortly be discussed.

Computing the marginal probability of a query is done by first grounding the program with respect to the query using backward reasoning. Afterwards, the ground logic program is rewritten into a logic formula that defines the truth value of the query in terms of the truth values of probabilistic facts. In the knowledge compilation step, the logic formula is transformed into a SDD which allows for more efficient WMC. SDDs support polytime model counting, conjunction, disjunction and negation. In the last step, the SDD is transformed into an Arithmetic Circuit (AC) which is evaluated in a bottom-up fashion by WMC. (Manhaeve, Dumančić, et al., 2021)

5.3. ProbEC

Prob-EC is a probabilistic extension of the Event Calculus (Kowalski and Sergot, 1985) and is based on the aforementioned ProbLog. The Event Calculus is a formalism for representing and reasoning with time. We have adapted the Prob-EC implementation of oPIEC [1] as it uses an online version of Prob-EC (Mantenoglou et al., 2023). The Event Calculus consists of time points T, events E and fluents F. Fluents are events which can be initiated and terminated. The main predicates of the Event Calculus are summarised in table 5.1.

[1] The corresponding code can be found on https://github.com/Periklismant/oPIEC/tree/main

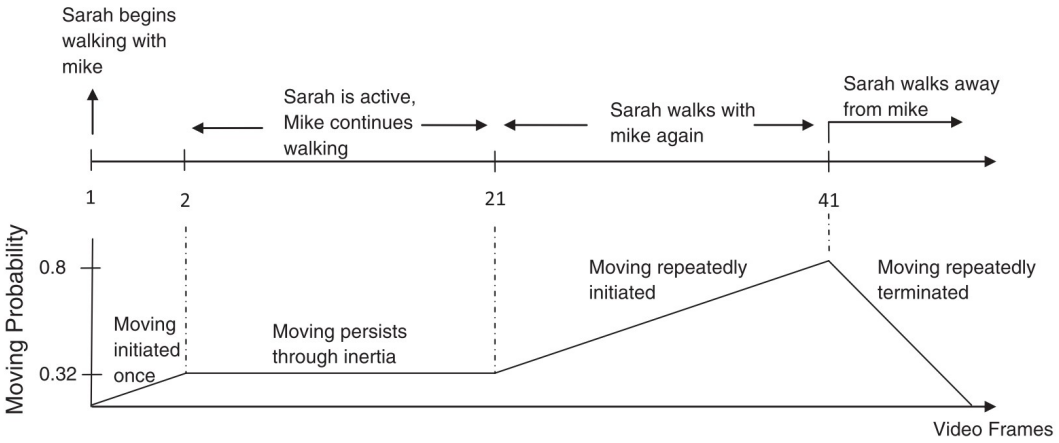


Figure 5.1: From Skarlatidis et al., 2015, example of the influence of multiple initiations and terminations on the probability of the complex event `moving`.

What makes the Event Calculus interesting is how it has integrated the common-sense law of inertia. The law of inertia states that an event holds at a certain time point if it has been initiated at a previous time point and not terminated in the meantime. Prob-EC has integrated the law of inertia in its probabilistic framework as follows: the probability of a complex event being true at time point T is equal to the probability of the disjunction of the initiation conditions of that complex event at a time point before T, assuming it has not been broken in the meantime. This leads to the following expression:

$$P(\text{ holdsAt } (CE = \text{true}, t)) = P\left(\vee_{\forall t_s < t} (\text{ initiatedAt } (CE = \text{true}, t_s) \wedge (\neg \text{broken } (CE = \text{true}, t_s, t)))\right) \tag{5.3}$$

This means that multiple initiations of the complex event increases its probability while terminations of the complex event (thus being broken) decreases its probability. An example is shown in Figure 5.1. When the complex event `moving` is repeatedly initiated, the probability of the complex event happening increases. When the complex event is not being initiated nor terminated, the probability stays constant and when the complex event is terminated the probability decreases.

5.4. DeepProbLog

DeepProbLog is a neurosymbolic framework that is also built on top of ProbLog. The key extension is the neural predicate or neural annotated disjunction (nAD). nADs are predicates that integrate neural networks whose outputs are finite probability distributions. Due to this extension, all essential components of ProbLog such as the semantics, the inference mechanism and the implementation are retained (Manhaeve, Dumančić, et al., 2021). More importantly, since both the probabilistic logic and the neural networks are differentiable, the entire program can be used for gradient-based training with a clear optimisation criterion namely the probability of the training examples (Manhaeve, Dumančić, et al., 2021).

A nAD has the form: $nn(m_q, [X_1, ..., X_k], O, [y_1, ..., y_n]) :: q(X_1, ..., X_k, O)$ where nn indicates the start of a nAD, m_q is a neural network identifier, $[X_1, ..., X_k]$ is the input to the neural network and O, the output of the neural network, is the probability distribution over the (discrete) domain $[y_1, ..., y_n]$. The neural network must be a discriminative classifier which can be considered a limitation. Similar to ADs, the distribution of probabilities from the output is mutually-exclusive over a set of clauses. When given specific input i, the probability distribution of the neural predicate is determined as $nn(m_q, i, y_1) :: q(i, y_1); ...; nn(m_q, i, y_n) :: q(i, y_n)$ where $nn(m_q, i, y_j)$ represents the probability that the neural network outputs y_j on input i. A well known example for an nAD is the classification of MNIST digits into their corresponding values:

$$nn(mnist_net, [X], Y, [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]) :: digit(X, Y)$$

When grounded, this nAD thus returns 10 probabilities, one for every possible number in the output domain $[0, ..., 9]$. The output layer of the neural network needs to be normalised. A common way to do

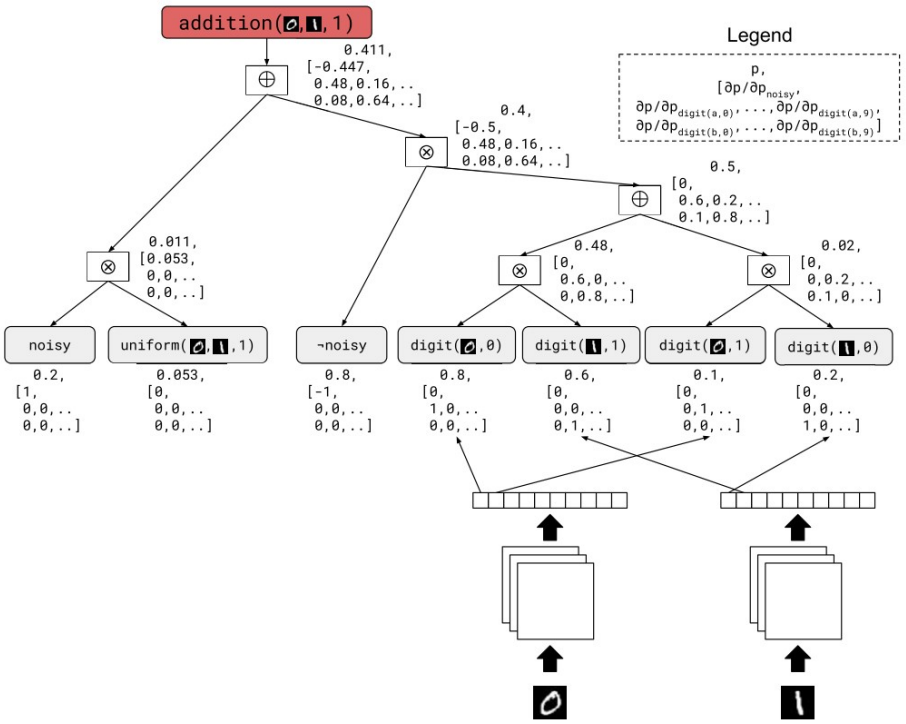


Figure 5.2: From Manhaeve, Marra, and De Raedt, 2021, parameter learning in DeepProbLog.

this is by adding a softmax layer.

Inference in DeepProbLog is almost identical to inference in ProbLog. The difference happens in the instantiation of the nADs. During grounding, the nADs are grounded and represent a symbolic representation of the probabilities which are then exchanged for their true value by making a forward pass on the relevant neural network. The success probability of the query is then computed by using these true values for a bottom-up evaluation of the AC. (Manhaeve, Marra, and De Raedt, 2021)

DeepProbLog makes use of discriminative training also known as learning from entailment (Manhaeve, Dumančić, et al., 2021). Since all elements in the DeepProbLog framework are differentiable, we can make use of gradient based learning. ProbLog makes use of its generalisation, Algebraic ProbLog (aProbLog), to not only compute the probabilities but also the gradient using the gradient semiring. This allows us to calculate the partial derivative of the success probability with respect to a parameter $\partial P(q)/\partial p_i$ which in turn can be used to perform backpropagation on the parameters of the neural network. This happens in two steps: first the gradient is propagated to the leaves of the AC, after which these gradients are propagated to the weights in the neural network. An example of an AC with partial derivatives with respect to some parameters is depicted in Figure 5.2. The bottom of the AC represents the inputs from the neural networks that take an image of a digit as input. The top of the AC represents the query. This way we can use weak supervision by training the neural networks with only the outputs of the entire DeepProbLog model. An overview of DeepProbLog learning is given in Figure 5.3.

This chapter has explained the different PLP frameworks being used in this research. This lays an important foundation for the theoretical discussions later on in the thesis.

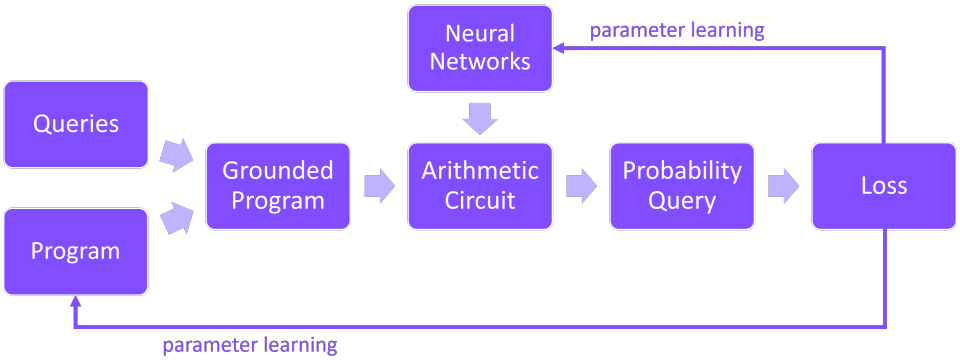


Figure 5.3: Overview of DeepProbLog learning.

6

Proposed Approach: The Hellenwaheri Framework

This chapter will go through the design of the Hellenwaheri framework by treating the different building blocks and how they came into place. Our framework combines the reasoning and learning power of nPLP frameworks with the temporal reasoning capabilities of Prob-EC for online detection in real-world video data. To this end, DeepProbLog is extended with Prob-EC and pre-trained computer vision algorithms to translate low-level data to simple events. An overview of the system can be found in Figure 6.1. We end this chapter with a summary on the functioning of the framework.

6.1. Integrating DeepProbLog and Prob-EC

As the documentation on DeepProbLog and Prob-EC was scarce, we relied mainly on the publications (Skarlatidis et al., 2015; Manhaeve, Dumančić, et al., 2021), code (comments) and examples to find out the working and possibilities of these frameworks.

From Chapter 5, we know DeepProbLog and Prob-EC are extensions of ProbLog. DeepProbLog extends ProbLog with the neural predicate that allows information exchange between the neural network and the AC. Prob-EC extends ProbLog with the Event Calculus to be able to define and track complex events and their probabilities. Prob-EC takes simple events as input and therefore does not support subsymbolic data as input. It has been used in an online setting in Mantenoglou et al., 2023, therefore we will be using this version. It replaces the clause `terminatedAt(F=true, T)` by `initiatedAt(F=false,T)`.

The main challenge lies in translating Prob-EC partially from the ProbLog program to Python such that it can efficiently communicate with DeepProbLog while retaining all its functionalities. Prob-EC, as in Mantenoglou et al., 2023, is completely executed in ProbLog. This means the initiation of the program, the processing of the simple events at every time point, the caching of the probabilities of complex events at previous time points and the querying of the complex events itself are all executed in ProbLog. DeepProbLog requires a ProbLog string as input as well as a query. Furthermore, the simple events are obtained through the processing of the video frames and the result of the query is obtained from DeepProbLog. This means that all these processes have to be removed from the ProbLog program and integrated into the DeepProbLog model. This caused a slight alteration in the inference process. Prob-EC first grounds the complex events clauses before querying, while DeepProbLog grounds the query and program at once. As the framework works online, frames are being processed as they come in. A new frame is processed once the processing of the previous frame is terminated. The speed is thus determined by the processing speed and not the streaming speed.

6.2. Integrating Subsymbolic Information

An important feature of our framework is the ability to detect complex events from real-world data. This means raw data needs to be converted to symbolic knowledge with which the program can reason. A first challenge is what subsymbolic information should be translated to symbolic information and how. A second challenge is where and how do we integrate this information in the Hellenwaheri framework.

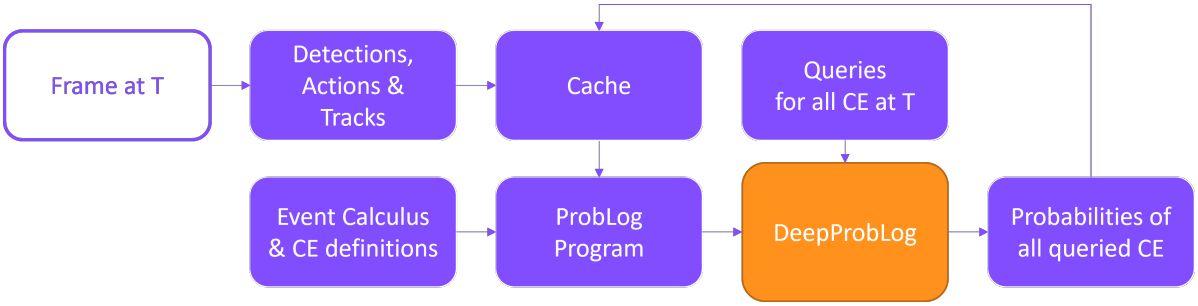


Figure 6.1: Overview of the Hellenwaheri framework designed in this research.

The first challenge has not been a focus point of the research since it is very context dependent and we are aiming for a proof-of-concept. Section 6.3 discusses the methods used in the proof-of-concept to translate the subsymbolic data into symbolic representations. The second challenge, however, does present some interesting choices. A logical step would be to integrate the neural networks with DeepProbLog with the use of the neural predicate. Unfortunately, the neural predicate only functions with discriminative classifiers which greatly limits the choice of neural networks to work with. For example, the position of an object or person is a valuable type of information but is not a discrete random variable. This means, such type of information cannot be integrated in the DeepProbLog model through the neural predicate. Naturally, if a type of simple events can be obtained with a classification network, it can be integrated as a neural predicate. For all the other cases, we need to find a different way to integrate the knowledge. The following options have been considered:

1. Integrate the knowledge as evidence. Evidence can be seen as a way to integrate conditional probability and has a similar set-up to queries. The program is grounded based on the evidence after which an AC is generated.

2. Add the knowledge to the ProbLog program before building the model.

3. Add the knowledge to the ProbLog program after the model is built.

A first glance at all options already shows a drawback: the information cannot be integrated after the AC has been compiled. This means the AC will have to be recompiled for every time point which has a high computational cost (Vilamala, 2022). Additionally, integrating the information before the AC is compiled causes a second drawback. These processes will not be part of the training loop and the framework will not be end-to-end differentiable for these neural networks. Keeping this is mind, all three options have been tested. Option 1 was not possible as evidence needs to be deterministically false or true. Probabilistic detections are a key component of our framework therefore we will not use the evidence feature. Option 2 is efficient as the program and model can be initiated at once. However, as new information is coming in at every time point, the model would have to be rebuilt every time point which is highly inefficient. Option 3 ensures the model is only initiated once but requires the program to be updated at every time point due to incoming information. This is computationally the best option, therefore we have chosen to implement option 3.

6.3. From Subsymbolic to Symbolic

6.3.1. Object detection and tracking

Previous section described how the simple events from low-level data could be integrated in the DeepProbLog model. Rest us now to know what knowledge these simple events will entail. We implement an object detector as well as an object tracking system to be able to track objects and persons over different frames. We use YOLOv7 (Wang et al., 2023) with pre-trained weights for object detection due to its speed and accuracy and SORT (Simple Online and Realtime Tracking) (Bewley et al., 2016) for 2D multiple object tracking. The candidate detections and their coordinates are passed on to SORT which returns the tracks and IDs of tracked entities. The class of the object, the location in pixels as well as the ID of the tracked object are translated to simple events. Furthermore, the average displacement in pixel per frame is calculated over 3 frames as an analogy for speed. With the exception of the class,

these values are continuous or infinite discrete and thus not the output of a discriminative classifier. As such, they have been integrated as described in section 6.2.

6.3.2. Human action recognition

An important part of the events that need surveillance are human bound. Human activities can be categorised into four different levels: gestures, actions, interactions and group activities (Karbalaie et al., 2022). Keeping the case of residential security in mind, actions and interactions are most interesting at this point. Gestures and group activities should of course not be discarded but are left as future research. Detecting interactions is simplified by using hierarchical knowledge reasoning. As discussed in section 2.2, instead of having to learn to detect an interaction from scratch, the different components of the interaction can be detected separately and together abstracted to form the interaction. For example, watching TV can be represented by its different components: a person sitting in front of a television. Finding a suitable human action recognition model is challenging for the following reasons:

- The model should be solely based on video imagery. Models that assume input data from inertial sensors or vision sensors, which is not available in the case of residential security, are not usable.
- Ideally, the model is incorporated as a neural predicate as human actions can be categorised into discrete variables (walking, running, sitting, lying down, etc.). This requires that the input to the model either comes from the available knowledge or can be deduced from a tensor variable as part of the query. A tensor variable is a variable that unifies with a tensor. For example, the neural predicate can take a tensor variable timestamp as input which can be deduced from the timestamp in the query. This timestamp can be used to obtain the frame (a tensor) of the input video corresponding to that timestamp on which the human action recognition model can be evaluated. A second requirement is that the human action recognition model should be end-to-end differentiable as the loss gradient can only be calculated if all components of the neural network are differentiable. This allows the model to be part of the training loop of the DeepProbLog model. Thus, DeepProbLog takes the input to the neural predicate, performs a forward pass on it for the given neural network and returns a probability for one of the output categories namely the human actions. Unfortunately, most action recognition systems are skeleton based meaning that the framework first translates the image of a person to keypoints that form a skeleton after which the keypoints are classified into an action. This process is not end-to-end differentiable.
- The model should classify simple human actions such as walk, run, sit, stand, etc. as interactions will be obtained from the rules. State-of-the-art human action recognition models are trained to recognise tens or hundreds of (complex) actions. This would unnecessarily complicate our model.
- Many models are trained with a camera at human eye level or lower. The different body parts are less distinct at a surveillance angle (higher up) which can make action recognition more challenging.
- Last but not least, the code needs to be publicly available to be incorporated in this framework. This is often not the case for state-of-the-art models.

Looking at the literature, simple events are often assumed as input such that low-level processing is not needed (Kardas and Cicekli, 2017; Mantenoglou et al., 2023) or human action recognition is simply not implemented (Caruccio et al., 2019; A. Khan et al., 2019). In Vilamala, 2022, an activity detection network is used but only returns if there is an activity in the video happening or not. The outputs of the neural network are limited to *activity* and *idle*. This is not expressive enough for our case. Acknowledging that meeting all requirements listed above is overly complicating this research, the human action recognition model will not be implemented as neural predicate. This gives us the possibility to use a skeleton-based human action recognition model. We combine a pose estimation algorithm[1] with a deep neural network action classifier (Chen, 2021). The actions with their corresponding IDs are then added as simple events to the ProbLog program as described in section 6.2.

Listing 6.1 shows an example of the simple events that have been obtained from a video for time point 30. In this example, the simple event `jump` is obtained from the human action recognition model,

[1] https://github.com/WongKinYiu/yolov7?tab=readme-ov-file#pose-estimation

entities such as `bicycle`, `car` and `person` and their coordinates `coord` are obtained from the object detector, and `avg_displ` is calculated from the coordinates and frame numbers.

```
processTimepoint(30):-
      assertz((0.52::happensAt(jump(id35),30))),
      assertz((0.76::happensAt(bicycle(id1),30))),
      assertz((1::holdsAtIE(coord(id1)=(3552,1169),30))),
      assertz((1::holdsAtIE(avg_displ(id1)=4,30))),
      assertz((0.32::happensAt(car(id5),30))),
      assertz((1::holdsAtIE(coord(id5)=(790,844),30))),
      assertz((1::holdsAtIE(avg_displ(id5)=4,30))),
      assertz((0.32::happensAt(motorcycle(id12),30))),
      assertz((1::holdsAtIE(coord(id12)=(819,1058),30))),
      assertz((1::holdsAtIE(avg_displ(id12)=3,30))),
      assertz((0.93::happensAt(person(id35),30))),
      assertz((1::holdsAtIE(coord(id35)=(2872,1651),30))),
      assertz((1::holdsAtIE(avg_displ(id35)=26,30))),
      assertz((0.76::happensAt(bicycle(id42),30))),
      assertz((1::holdsAtIE(coord(id42)=(213,1042),30))),
      assertz((1::holdsAtIE(avg_displ(id42)=1,30))).
```

Listing 6.1: Example of simple event inputs from the low-level data for time point 30. The red text represents the IDs to which the simple event applies, the number before the double colon represents the probability and the last number represents the time point.

6.4. Cache

From Chapter 5, we know that in Prob-EC the probability that a complex event holds at a time point depends on the probability that the complex event held previously and was not broken in the meantime. To be able to keep track of previous probabilities and thus for a correct functioning of Prob-EC, a cache system is necessary. The cache is integrated by adding it to the ProbLog program before building the AC. The cache can be seen as a ProbLog program where probabilistic facts are removed and added based on the time point. In fact, the cache will keep track of all knowledge that was not available at the start of the video, including the simple events from low-level data. Prob-EC, as integrated in oPIEC (Mantenoglou et al., 2023), uses a cache system where it saves the probability of the last query for a complex event. When that complex event is called again, the probability is compared with the current probability of the queried complex event and if the difference is larger than 0.0001, the new probability is saved. We use a similar set-up but with some alterations. The cache is maintained outside of the original ProbLog program. Furthermore, the full query is saved including the time variable. Before feeding the knowledge base into the DeepProbLog program, the cache is updated by adding new knowledge and removing superfluous knowledge. This is depicted in Figure 6.2. First, the simple events from the low-level data for that time point and the queried result from the previous time point are added. The result has the form `0.67::holdsAt(leaving_bag(id12, id8)=true, 40).` Secondly, by caching the results of the higher level complex events, the lower level data becomes superfluous. Therefore, all expressions associated with low-level data are removed from the cache after two timesteps. Lastly, a result of a complex event being true is either removed if that complex event is detected again with the same track IDs (thus being replaced) or after two time steps. These measures also ensure that the cache does not grow exponentially in time.

6.5. Querying

We are interested in computing the marginal probability of a complex event at a certain time point. This is the probability of the complex event being true over all possible worlds at that time point. This can be achieved by querying the DeepProbLog program for the complex event. Such a query consists of the predicate `holdsAt(F=V, T)` in which F is the complex event and V is the value of the complex event at time point T. For querying, V will always be set to `true`. An example of such a query is `holdsAt(leaving_bag(Person,Bag)=true,40)` that returns the probability of the complex event `leaving_bag(Person,Bag)` being true at time point 40. If we have multiple complex events we are

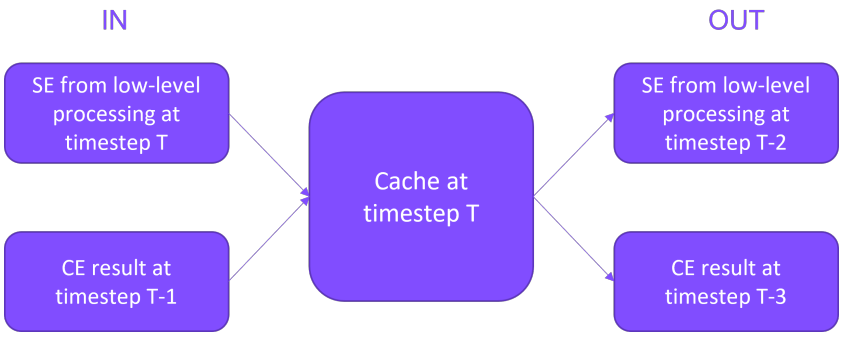


Figure 6.2: Update of the cache system at a time point T. (SE=simple events, CE=complex events)

interested in, the queries (one for every complex event) can be added in batch mode. DeepProbLog then grounds the program for every query that are then compiled into different ACs. This means an AC is compiled for every complex event at every time point. Unfortunately, ProbLog only supports queries that consist of a single term thus grounding for multiple queries at once is not an option. DeepProbLog does offer the possibility to cache an AC based on the query. However, as the time variable is included in the query, the query changes for every time point and the AC cache system can therefore not be used.

6.6. ProbLog Program

The ProbLog program consists of the Prob-EC clauses as well as the complex event definitions and the domain knowledge. The complex event definitions and the domain knowledge will be discussed in Chapter 7 as these are context specific. Prob-EC, however, remains the same for every use case, see Listing 6.2. When querying for a complex event, the probability of the disjunct of both `holdsAt(F=V,T)` definitions is calculated. The first represents the probability that the complex event has been initiated at an earlier time point, represented by the `cached()` predicate, and has not been broken in the meantime. The second represents the probability that the complex event has been initiated at the previous time point. For an example on how this influences the probability of a complex event, we refer to Chapter 5. The predicate `processTimepoint(T)` ensures that all the simple events from the low-level data saved in the cache are added to the program. This process is depicted in Figure 6.3.

```
% calculates disjunct F=V still holds.
holdsAt(F=V,T):-
  prevTimepoint(T,Tprev),
(  processTimepoint(Tprev),fail;true),
  cached(holdsAt(F=V, Tprev)),
  \+ broken(F=V, Tprev).

% calculates disjunct F=V is initiated.
holdsAt(F=V,T):-
  prevTimepoint(T, Tprev),
(  processTimepoint(Tprev),fail;true),
  prevTimepoint(Tprev, Tprevprev),
(  processTimepoint(Tprevprev),fail;true),
  initiatedAt(F=V,Tprev).

% F=V1 is broken when F is initiated withthe value false.
broken(F=V1, T):-
  initiatedAt(F=false,T).
```

Listing 6.2: Using my custom listings style

In summary, the framework works as follows (see 6.1 for an overview): the frame at time point T is used to obtain object detections and pose estimations. The object detections are forwarded to

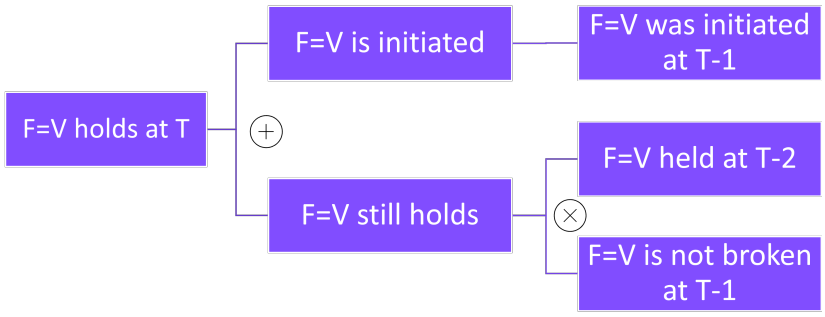


Figure 6.3: The probability that a complex event holds at time point T. The summation sign refers to a disjunction and the multiplication sign refers to a conjunction.

the tracking algorithm which couples the detections to an ID. The skeletons from the pose estimation algorithm are compared to the tracked detections and, if overlapping, coupled to the same ID. The information from these low-level processing algorithms is added to the cache which is added to the ProbLog program that contains the Event Calculus, the complex event definitions and the domain knowledge. The queries for the complex events and the ProbLog program are given as input to the DeepProbLog model for inference. DeepProbLog returns a probability for every complex event at time point T. The cache is updated and the process repeats for the next frame. This chapter has treated all the building blocks of the Hellenwaheri framework for the general case. The full code is available on request.

7

Experiment Design

The Hellenwaheri framework is designed for online complex event detection in real-world surveillance video data. We begin this chapter with explaining how the framework is adapted for this use case after which we describe how the framework is evaluated. All runs in this and the next chapter are performed on a NVIDIA GeForce RTX 2080 Ti GPU with 11.0 GB of GDDR6 memory.

7.1. Dataset

The dataset on which the framework is tested should be video footage, preferably from a surveillance angle, of suspicious events. Initially, the CAVIAR dataset[1] was chosen due to its popularity in complex event detection researches (Qiu et al., 2020; Skarlatidis et al., 2015; Vilamala, 2022), its surveillance application and the availability of the simple events at every time point. However, a quick test with YOLOv7 on the CAVIAR dataset shows why CAVIAR is undesirable for this research (see Figure 7.1). In only a few frames, a person was correctly detected while in many frames, there was either no detection or wrong detections such as birds and skateboards. Essential objects such as a bag were not detected at all. We suspect this is due to the low quality of the video images of the dataset (384x288) and the angle that differs from the COCO dataset on which YOLOv7 has been trained. A quick scroll through the COCO dataset shows most of the images have been taken on human eye level. Looking at the literature, few have tried to use object detection on CAVIAR and use image processing techniques rather than deep learning. As this is not scalable, we will not be using CAVIAR for the research. A quick search into other suitable datasets were not conducive to better results. With the following points in mind, we have decided to record our own dataset:

- Since the goal of this research is not to train the model (although this is theoretically possible), there is no need for a large dataset. In contrary, a few videos with suspicious actions can prove the concept.
- Videos need to be recorded with a static camera, contain human actions preferably in surveillance context (interactions with objects) and be of good enough quality as we will not be focusing on improving detection algorithms.
- The results of this research should not depend on the performance of the processing of the computer vision tasks. This is only a part of the full pipeline and can easily be improved in the future. Either by selecting algorithms that perform better on the task at hand (algorithm trained on images from surveillance angle) or by using available information to improve detections (camera calibration parameters).

The dataset consists of several short videos in which a suspicious event happens and one longer video in which multiple suspicious events happen. The videos have been recorded in such a way that the objects and actors are clearly visible, there are no occlusions and the weather is good (for Dutch standards). The videos have a 4k resolution (3840 x 2160 px) with 30 frames per second (fps). Figure 7.2 shows two frames from the dataset.

[1] https://homepages.inf.ed.ac.uk/rbf/CAVIAR/

Figure 7.1: YOLOv7 detections of the leaving bag scene in the CAVIAR dataset

Figure 7.2: Two frames from the recorded dataset.

7.2. Computer Vision Layer

The object detection and tracking algorithm have been slightly adjusted to the use case. The classes used from the object detector are person (0), bicycle (1), car (2), motorbike (3), backpack (24), handbag (26) and suitcase (28). Furthermore, for a more consistent detection, the different types of bags (backpack, handbag and suitcase) are returned as one new class 'bag'. The object detection framework will not be trained, YOLOv7 is used with pre-trained weights. The human action recognition pipeline is integrated as follows:

1. Skeletons are obtained from the input image using the pre-trained weights of YOLOv7 for pose estimation [2]. An example can be seen in Figure 7.3.

2. The action classifier requires the 18 keypoints OpenPose notation [3] for the skeletons. This requires a different order of the keypoints as well as an extra keypoint between the shoulder joints.

3. The skeletons have a confidence score attached for every keypoint in the skeleton. Keypoints with a confidence lower than 0.5 are removed and skeletons with less than five keypoints (with a confidence score above 0.5) are filtered out.

4. For every remaining skeleton, a bounding box is calculated. The skeleton is given a track ID if the overlap between the bounding box of a detected person with that track ID and the bounding box of the skeleton is highest for that skeleton. Furthermore, the overlap should be more than 0.5. This way, only tracked skeletons are kept.

5. The filtered and tracked skeletons of 5 consecutive frames are given as input to the action recognition model (partially taken from Chen, 2021) and classified into one of the following 7 classes: `stand`, `walk`, `run`, `jump`, `sit`, `punch`, `wave`. The classes `kick` and `squat` from the original 9 classes have been replaced with `walk` and `sit` respectively as these were often confused. The algorithm works as follows: first, the skeleton data is preprocessed, then features of

[2] https://github.com/WongKinYiu/yolov7?tab=readme-ov-file#pose-estimation
[3] https://www.researchgate.net/figure/The-body-parts-of-skeleton-model_fig1_329898289

Figure 7.3: Example of the pose estimation algorithm used for human action recognition.

the skeleton are generated which are fed into the deep neural network that classifies the action. For a complete overview of the functioning of the algorithm, see Chen, 2021.

7.3. Medium-Level Event Definitions

With the simple events obtained from the low-level data, we can derive multiple medium-level events that in turn can be used for the definitions of the complex events. These type of events are characterised by the fact that they cannot directly be obtained from the low-level data but are also not being queried. `avg_displ` is a special type of medium-level event as it is integrated as a simple event. This is due to the dependence on multiple frames which cannot be achieved within ProbLog rules with the current set-up. We will shortly discuss the medium-level events `close` and `moving_closer` used in this research. The medium-level event clauses are defined in such a way that probabilities are returned instead of a boolean value. This ensures that the inference process is not stopped at once when returning `False` but returns a very low probability. This is also more intuitive as the probability can be used as a continuous scale to indicate the degree of the medium-level event being true. This can be seen as a way to deal with pattern uncertainty, i.e. the uncertainty in the event definitions (see Section 3.2).

The distance between two entities can be obtained from the coordinates of these entities. With enough information such as camera calibration parameters, depth information or known object dimensions, this distance can be calculated in real-world dimensions. However, for the current set-up uses pixels as distance measure for simplification. Furthermore, if the Marechaussee uses static cameras with known calibration parameters, this type of information can easily and accurately be obtained. From the distance between two objects, one can determine if these objects are close or not. The medium-level event `close` is integrated probabilistically, see Listing 7.1. The closer the two entities are, the higher the probability of the medium-level event `close`, represented by `Prob` in Listing 7.1 (line 1-3). The probability is weighed by a parameter `D`, that is dependent on the context (see Section 7.5). The clause `calculateDistance` returns the distance in pixels between two entities. The distance between the two entities is subtracted with 100 pixels as this is approximately the minimum distance possible. The probability that `close` is false is calculated in a similar way, however, the probability now grows larger when the distance between the two entities increases (line 5-7).

```
1 Prob::holdsAtMacro(close(Person1, Person2, D) = true, T) :-
2    calculateDistance(Person1, Person2, T, Dist),
3    Prob is max(0,(1-max(0,(Dist-100))/D)).
4
5 Prob::holdsAtMacro(close(Person1, Person2, D) = false, T) :-
6    calculateDistance(Person1, Person2, T, Dist),
7    Prob is min(1,(max(0,(Dist-100))/D)).
8
```

```
calculateDistance(Person1, Person2, T, Dist):-
    \+ Person1 = Person2,
    holdsAtIE(coord(Person1) = (X1, Y1), T ),
    holdsAtIE(coord(Person2) = (X2, Y2), T ),
    XDiff is abs(X1-X2),
    YDiff is abs(Y1-Y2),
    SideA is XDiff*XDiff,
    SideB is YDiff*YDiff,
    Temp is SideA + SideB,
    Dist is sqrt(Temp).
```

Listing 7.1: The medium-level event `close` is probabilistically true with a weight determined by the distance in pixels between two entities

The probability for the medium-level event `moving_closer` is obtained by calculating the distance between two entities at two consecutive time points, and using the change in distance from the first time point to the second time point as a weight for the probability. If the change in distance is high the probability of `moving_closer` will be high. If the change in distance is negative or zero, the probability of `moving_closer` will be zero. Listing 7.2 shows the clause for `moving_closer`. For this case, we do not need `moving_closer(Person1, Person2) = false`.

```
Prob::holdsAtMacro(moving_closer(Person1, Person2) = true, T):-
    closingDist(D),
    prevTimepoint(T, Tprev),
    calculateDistance(Person1, Person2, Tprev, Dist1),
    calculateDistance(Person1, Person2, T, Dist2),
    DeltaDist is max(Dist1-Dist2,0),
    Prob is min(1,DeltaDist/D).
```

Listing 7.2: The medium-level event `moving_closer` is probabilistically true with a weight determined by the change in distance between two entities. For the definition of `calculateDistance`, see Listing 7.1.

7.4. Complex Event Definitions

For every complex event that we are trying to detect, the rules initiating and terminating that complex event should be defined. Here, we present two examples of how these rules can be defined. For a complete overview of the rules, we refer to Appendix A. The complex events we are trying to detect are `leaving_bag` and `holding_bag`. Their definitions are shown in Listing 7.3. The complex event `holding_bag` is initiated when a person and a bag are detected close together (line 3-6). The medium-level event `close` is true with a probability that is inversely proportional to the distance between person and bag (line 6). The complex event is terminated when the complex event `leaving_bag` is initiated (line 13). `leaving_bag` is initiated when a person that was holding a bag in a previous frame, is not close to that bag anymore (line 17-22). This probability is now directly proportional to the distance between the person and the bag (line 22). The complex event is terminated when either the person disappears (line 26-28) or the person is close to the bag again (line 30-33). Together with the Event Calculus as described in Chapter 6 and some other practical rules, the complex event definitions form the ProbLog program used for inference in DeepProbLog.

```
% ----- initiate holding_bag

initiatedAt(holding_bag(Person,Bag) = true, T):-
    happensAt(person(Person), T),
    happensAt(bag(Bag), T),
    holdsAtMacro(close(Person, Bag)=true, T).

% ----- terminate holding_bag

initiatedAt(holding_bag(Person,Bag) = false, T):-
    happensAt(person(Person), T),
```

```
12    happensAt(bag(Bag), T),
13    initiatedAt(leaving_bag(Person, Bag) = true, T).
14
15 % ----- initiate leaving_bag
16
17 initiatedAt(leaving_bag(Person, Bag) = true, T):-
18    happensAt(person(Person), T),
19    happensAt(bag(Bag), T),
20    prevTimepoint(T, Tprev),
21    cached(holdsAt(holding_bag(Person, Bag) = true, Tprev)),
22    holdsAtMacro(close(Person, Bag)=false, T).
23
24 % ----- terminate leaving_bag
25
26 initiatedAt(leaving_bag(Person, Bag) = false, T):-
27    happensAt(bag(Bag), T),
28    \+ happensAt(person(Person), T).
29
30 initiatedAt(leaving_bag(Person, Bag) = false, T):-
31    happensAt(bag(Bag), T),
32    happensAt(person(Person), T),
33    holdsAtMacro(close(Person, Bag)=true, T).
```

Listing 7.3: The definitions of the complex events `leaving_bag(Person,Bag)` and `holding_bag(Person,Bag)`

DeepProbLog returns the probabilities of the queries being true. We can attach a confidence threshold to the output that signifies an alarm going off. If the probability of a complex event happening surpasses the threshold, the workforce is alarmed. This threshold is determined after the first experiments.

7.5. Domain Knowledge

Domain knowledge is grounded knowledge about the scene or use case that is manually added by the domain expert. Assuming residential security is performed with static cameras increases the ease of implementing domain knowledge. For example, certain areas can be marked as alarming or using camera calibration parameters, the depth of entities can be calculated. In our case, the domain knowledge mainly translates to threshold parameters. These parameters are statistically determined and given to the model in the form of facts. These facts are then used for the abstraction of simple events (such as centre of the bounding box) to medium-level events (such as `close`, `moving_closer` or `same_speed`). The threshold parameters and their meaning are listed in Table 7.1.

7.6. Evaluation

Expressivity, efficiency and adaptability together encompass the challenges and practicalities that come with this application. The framework will be assessed quantitatively and qualitatively on how it fulfils each of the requirements such that a complete and sound conclusion can be formed that answers the research question. During this evaluation, one should keep in mind that this research delivers a proof-of-concept for using this type of framework on complex event detection and will therefore not be compared to other state-of-the-art neurosymbolic models for complex event detection. The adjectives will now be explained one by one.

7.6.1. Expressivity

Simply said, expressivity in this research signifies the ability of the framework to represent a wide range of concepts. We do not want the workforce to be limited in what type of suspicious behaviour can be detected around the residence. Furthermore, expressivity also looks at how well the low-level data is abstracted to symbolic knowledge. Part of this looks at the effect of uncertainty on the model. Lastly, it looks at the performance of the model in terms of the accuracy of the detections of the different complex events.

Parameter	Value	Meaning
`holdDist`	600	The maximum distance, in pixels, between an object and a person for it to be considered that the person is holding the object.
`meetingDist`	400	The maximum distance, in pixels, between two persons for them to be considered as meeting.
`closingDis`	40	The minimum change in distance, in pixels, over two frames between two entities for them to be considered as moving closer together.
`maxSpeedDif`	40	The maximum difference in speed, in pixels per frame, between two entities for them be considered having the same speed.
`walkDist`	800	The maximum distance, in pixels, between two persons for them to be considered as walking together.
`abruptDispl`	200	The minimum displacement, in pixels per frame, of an entity for it to be considered as doing an abrupt motion.
`stationDispl`	2.5	The maximum displacement, in pixels per frame, of an entity for it to be considered as stationary.
`movingDispl`	10	The minimum displacement, in pixels per frame, of an entity for it to not be considered as stationary anymore.

Table 7.1: Domain knowledge for the use case of surveillance. The parameters are based on a frame with size 3840x2160 px.

7.6.2. Efficiency

For this research, we will solely focus on time-efficiency and not data-efficiency. The second could be interesting in a training setting when compared to different frameworks. We leave this for future research. Time-efficiency represents the time required to prepare the system and the time required for inference in different scenarios. We will time the processes of the framework for different settings to assess the influence of different factors on the efficiency and thus speed of the framework. As mentioned earlier the system operates online and not necessarily real-time. However, it is insightful to know to what extent the framework could operate real-time. Furthermore, Chapter 6 already pointed out some unsatisfactory adaptations in terms of the order of processing. These will be discussed in more detail when analysing the inference efficiency of the framework.

7.6.3. Adaptability

Adaptability refers to how the framework should be adapted such that it is ready for operation. As the framework is designed to be a proof-of-concept, design choices have been made to simplify the detection process. We will touch upon the steps that need to be taken for the framework to function properly as an IVS system and be used by the workforce. Furthermore, the adaptability to different applications will be discussed as well. This will give insight into the robustness of the framework.

In summary, the framework will be tested on a newly recorded dataset of suspicious events. The framework should return the probability of a complex event happening at a certain time point. The final evaluation is based on expressivity, efficiency and adaptability.

8

Improving Inference

This chapter looks into the different possibilities to improve the inference speed of the Hellenwaheri framework. Although knowledge compilation already increases the inference speed, the DeepProbLog inference mechanism is still inefficient for complex event detection in general and more specifically in the Hellenwaheri framework. Following Manhaeve, Marra, and De Raedt, 2021, there are two computational problems in DeepProbLog inference. The first being the need to compute the set of all possible worlds which can explode and the second being the compilation into an arithmetic circuit which can be computationally hard. The first problem is also part of the second problem as compilation becomes simplified and faster by using a subset of all possible worlds for inference. Compiling the AC is the most costly process of DeepProbLog inference (Vilamala, 2022). We therefore focus on reducing the compiling time of the AC. The importance of this problem is affirmed by Vilamala. From our point of view, this can be achieved in two ways. Either the AC should be compiled less often or the compilation time per AC should be reduced. Both these options will extensively be discussed in this chapter. The process of compiling the AC refers to all the steps from grounding the program and the query to forming the SDD (see Section 5.2 for more information).

8.1. Reduce Compiling Frequency

By ensuring the AC is not compiled for every time point, efficiency can be increased. This problem has been approached in the literature in different ways. DeepProbLog has a feature that is able to cache the ACs based on the query. In Vilamala, 2022, the cache feature increases the speed of the program up to 4 times. In the example of calculating the sum of the MNIST digits, an AC can be obtained from the cache if the addition that is queried has been queried before. The query `addition(MNIST1,MNIST2,Sum)` is used as the key to the cached AC where `MNISTi` represents the tensor of an MNIST digit image. For an AC to be re-used, the exact same image for both inputs as well as the exact same number for the sum should be used. It is important to realise that using the cache feature of DeepProbLog only speeds up inference if the exact same query is used multiple times. Looking at complex event detection with the Hellenwaheri framework, the query is never twice the same due to the time variable in the query. DeepProbLog offers the possibility to replace a variable in the query after compilation if the variable is used as input for the neural predicate. This is unfortunately not the case in the Hellenwaheri framework. The downside of the cache system is the memory needed to cache the ACs. For larger datasets, this can be a limiting factor.

Another approach used to reduce the compiling frequency is PreCompilation (Vilamala, 2022). PreCompilation was designed in the context of temporal relations which introduces a time variable in the query. The framework allows the re-use of an AC for different time points. It makes use of the fact that the query only changes due to the time variable and adjusts the program such that it is independent of the time variable. The AC then only has to be compiled once for every type of query. PreCompilation replaces the time variables in the program by constant identifiers that are defined in relation to the time point in the query. This means the size of the AC is constant and only the weights of the nodes (the probabilities) are changed. When the program is queried at a certain time point, the AC with identifiers for that type of query is re-used by changing the probabilities of the nodes for the inputs corresponding

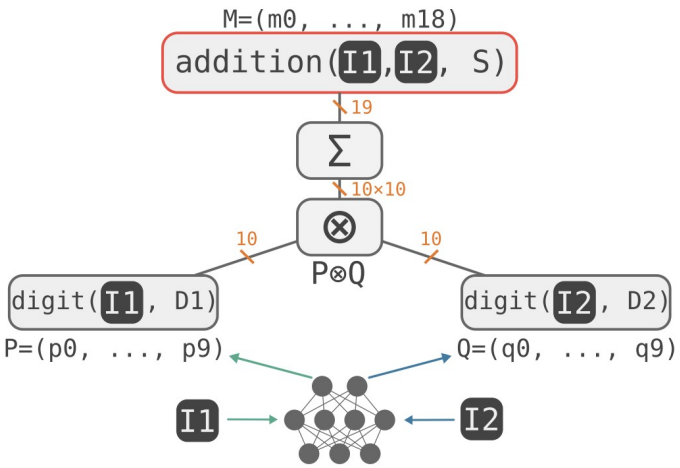


Figure 8.1: From Smet et al., 2022, a tensor-lifted AC for the addition query with tensore variables. The branches are annotated with the dimensionality of the signal that they carry.

to that time point. This points out a first limitation. The AC needs to be compiled such that all the inputs or simple events that can arise are nodes integrated in the AC. Vilamala performs an experiment in which the query `twoHeads(T)` is true if the inputs `heads1(T)` and `heads2(T)` are true. In this case, there will always be two discrete inputs, one for every coin. The AC is then compiled with these two inputs as nodes of which the probability is changed for different time points. In many cases, the simple events are not known beforehand which prevents us from integrating them in a general AC. For example, when obtaining the simple events from real-world video data, we do not know how many objects will be detected and thus how many simple events we are dealing with. The framework does work recursively by creating a node in the AC for the probability of the query at a previous time point. The probability of this node can then be changed at every time point and thus works just like a cache system. PreCompilation has, however, not been integrated with a neurosymbolic framework yet.

The last approach we will be discussing is using a tensor-lifted AC such as in Smet et al., 2022. This idea is related to lifted reasoning, a promising research direction for improving inference efficiency in multiple applications. Lifted reasoning is performed on the level of the original non-grounded program thus with variables. A tensor-lifted AC exploits the symmetry in a neural probabilistic logic program that uses neural predicates to facilitate parallelisation. The AC is compiled with a non grounded query allowing us to re-use the AC for all values of the input variables. However, this only applies to query variables that unify with a tensor, or tensor-variables as De Smet calls them. This means the AC is only compiled once for any set of input tensors. From the domains of the neural predicates, all necessary domains of target tensors can be calculated. The inference process to arrive at these target tensors is very similar for different tensor values which would result in a very repetitive AC. This symmetry can be used to tensorise the conjunctions and disjunctions of tensor variables, such that one tensor-lifted AC can be compiled that only requires grounding once. An example of this type of AC is displayed in Figure 8.1. Once again, we use the example of performing an addition on the MNIST digits where the sum of the inputs is tensorised. The probability of the query is calculated by taking a tensor product (representing the probabilities of all combinations of two MNIST digits) and summing over the results that lead to the same value in the target domain (0 to 18).

We now look at how this relates to the application of complex event detection. The variables in the query as used in this research are not tensor variables. If we would, hypothetically, create an AC for all time points and inputs, the repetition would be found in the inference of the complex event definitions for different time points. To exploit this symmetry such that a tensor-lifted AC can be used, time should be made a tensor-variable. The reason why the variable has to be a tensor variable is due to its independence of other nodes. When looking at an AC, the probabilistic facts and neural predicates are on the leaves of the AC. Attaching a probability to these nodes represents the start of the bottom-up

inference process. This means the structure of the AC itself doesn't change when replacing tensor variables, only the probabilities of the leaf nodes do. If we want time to be a tensor-variable it should be used as input to the neural predicate. In principle, the time variable could be used to obtain the frame from the video corresponding to that time point. This set-up could work if all the simple events needed to answer the query are obtained through neural predicates. However, this is not possible at this point in time. Furthermore, the tensor-lifted AC takes all possible values into account. In the case of having 10 digits and one tensorised inference step with 19 possible outputs, this doesn't necessarily create a problem. However, tensorising the logic inference for all complex event definitions and Prob-EC clauses combined for all the possible inputs would be more problematic. It is worth questioning if the inference speed would increase.

We have also looked into extending the AC instead of compiling from scratch but this did not provide any results. Lastly, we have looked at ways to compile one AC for multiple queries but this is not supported by DeepProbLog.

8.2. Reduce Compiling Time

Exact inference such as in DeepProbLog is a time-consuming process. Approximate inference could reduce the computational complexity for the inference process. One line of research in approximate inference tries to find ways to only compute a subset of proofs instead of all proofs which could reduce compilation time (Manhaeve, Marra, and De Raedt, 2021). Examples are k-best proofs (Renkens et al., 2012) and DPLA* (Manhaeve, Marra, and De Raedt, 2021). In k-best proofs, only k proofs ($P(L)$ in Equation 5.2) with the highest probability are used to calculate the approximate success probability of the query. This calculates a lower bound on the success probability. DPLA* is an extension of k-best proofs by using different heuristics. For example, a heuristic that is the estimate of the probability of a partial proof (see Manhaeve, Marra, and De Raedt, 2021 for a more information). Naturally, training is also affected by approximate inference as inference is used to obtain the output used for training. One challenge for approximate inference in neurosymbolic models is the fact that the best proofs are chosen based on the probabilities of facts and neural predicates which are usually not correct at the beginning of the learning trajectory. DPLA* uses curriculum learning and a form of exploration to overcome this. Manhaeve, Marra, and De Raedt, 2021 also proves that approximate inference is faster with pre-trained networks and increased neural predicate accuracy (classification accuracy). The accuracy is slightly lower than with exact inference but the scalability increases. DPLA* has been integrated in DeepProbLog. We expect that approximate inference will not considerably influence the compiling time of the AC as implemented in the Hellenwaheri framework. Due to hierarchical set-up of the program, the small number of probabilistic facts and the absence of neural facts, the number of proofs will be small in any situation. Approximate inference could become interesting when all simple events are obtained through neural predicates. Neural predicates may introduce many new nodes and thus inference directions as the neural predicate is evaluated on every value in the output domain that is consistent with the logic program. Approximate inference could ensure not all these inference directions are being followed.

8.3. Proposed Solution

It becomes clear that we are greatly limited by the fact that not all simple events are obtained through the neural predicates. Let us assume that due to new developments in the future, the simple events are obtained through neural predicates. Reanalysing the solutions presented earlier based on this new information yields two interesting possibilities to improve the inference: caching of the ACs and approximate inference using DPLA*. DPLA* has already been integrated in DeepProbLog and does not require any adaptations for our application. To be able to use the caching, the query should remain the same for different time points. We propose a method in which the query does not explicitly contain a time point but a tensor variable. This means, assuming the ProbLog program remains the same, the AC does not have to be recompiled as we can query for all complex events at once. Using the substitution function of DeepProbLog allows us to substitute a variable in the query with a tensor after compilation. In this case, the substitution would contain the different time points. This means that the time points attach probabilities to the leaves of the AC (the simple events). A new system for the cache will have to be designed as the cache is now integrated in the knowledge base before compilation. A challenge that we expect is the fact that the cached complex events are grounded as opposed to the

queries. Currently, a query has the form:

```
holdsAt(leaving_bag(Person,Bag)=true,230)
```

while a cached complex event has the form:

```
cached(holdsAt(leaving_bag(id14, id81) = true, 230))
```

As we do not know what objects or persons will be detected, thus not knowing the IDs, the AC will not be able to have a node for the grounded cache. A non-grounded cache will be ambiguous in what cached complex event it refers to.

The practicality of this solution remains to be tested and is left as future research.

9 Results

The results of testing the performance of The Hellenwaheri framework are reported in this chapter. We will first look at the performance of the computer vision layer that transforms the low-level data to simple events. This layer is mainly responsible for providing the probabilistic facts with which the model reasons, therefore it is important to analyse these results separately. We then continue this chapter with the results of the full framework. We provide several examples of complex events that can be defined for this use case. This will give us an indication of how well the framework can detect the different complex events. Lastly, we look at the time-efficiency of the framework for different parameters.

9.1. Abstracting Low-Level Data

The lower level processing uses four different algorithms: the detection algorithm, the tracking algorithm, the pose estimation algorithm and the action recognition algorithm. All four will be assessed visually in this section.

The object detection algorithm takes a frame as input and returns the class, probability and location of the detections. These detections are forwarded to the tracking algorithm that assigns an ID to them if they appear in a number of consecutive frames. The results of the detection and tracking algorithm are analysed by displaying the bounding box of the detections and the tracks over the frames. If not stated otherwise, all results in this chapter have been obtained with a fps rate of 10 and a lower resolution of 640x360 as higher resolutions only degraded the tracking performance. This is probably due to the fact that YOLOv7 was trained with a resolution of 640. Figure 9.1 shows the result for 4 frames, i.e. 2 frames per video. The results are satisfactory for multiple reasons. First, the tracking algorithm manages to keep track of the main persons and objects in the video. This can be concluded from the ID given to the entities, they do not change over time. Even when the person is crouched, such as in the right lower image, the person retains its ID. Secondly, the bounding boxes are relatively accurate. In most cases the bounding box correctly frames the entity as a whole. Figure 9.3 displays a case where the bounding box frames a different part of the entity which causes a sudden change in the centre coordinates of the entity. These coordinates are translated to the simple event `coord` which is then used to form medium-level events. This way, a small artefact in the computer vision layer can influence the detection of the complex events. In Figure 9.2, we see that the IDs of the two persons have been switched. This can happen when the persons come too close to each other. The tracking algorithm uses basic data association principles and does not handle these scenarios well. Thirdly, the probabilities associated with the detections are often higher than 0.8. This is favourable for the inference process as performing inference with higher probabilities leads to a more confident detection for the complex event.

The pose estimation algorithm takes a frame as input and returns a skeleton for every person it detects. The skeletons can be drawn onto the frame to analyse the results. The different segments of the skeleton are mostly accurately overlapping the corresponding body parts. Even when a part of the body is occluded, the pose estimation algorithm manages to calculate the missing segments. Examples of this can be seen in Figure 9.4. We conclude the algorithm has a satisfactory performance.

The action recognition algorithm takes the skeletons in OpenPose format as input and returns an action for every skeleton (see Figure 9.5. If the probabilities of all the classes are under a manually

Figure 9.1: Detections and tracks for multiple frames.

Figure 9.2: IDs of the tracked entities are switched between the persons in this video. The images have been cropped compared to the original frame.

Figure 9.3: Example of bounding boxes that frame different parts of the entity, in this case a person. This results in a sudden change in the simple event `coord` that keeps track of the centre of the bounding box.

Figure 9.4: Examples of the output of the pose estimation algorithm when part of the body is occluded. The images have been cropped compared to the original frame.

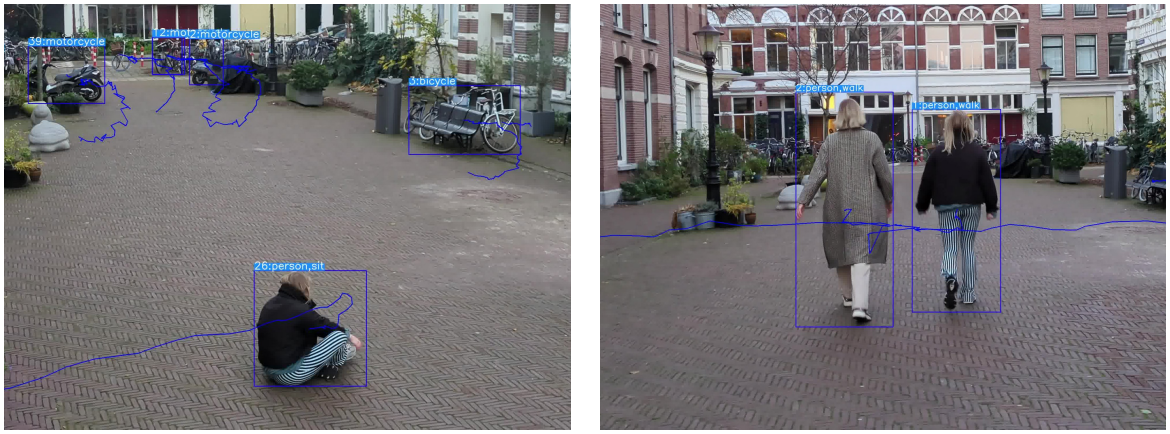


Figure 9.5: Examples of the output of the action recognition algorithm. The images have been cropped compared to the original frame.

defined threshold, the model returns an empty string. The performance of the action classifier is disappointing, especially considering the accuracy of the pose estimation algorithm. Replacing the classes `kick` and `squat` by `walk` and `sit` respectively gives slightly better results but the performance still remains unsatisfactory. Therefore, we are careful with integrating the simple events generated by the action recognition algorithm in the medium-level or complex event definitions. The action `stand` is only recognised when a person does not move for multiple frames with the arms next to the body. For an example of wrong classifications, see Figure 9.6.

9.2. Complex Event Detection

We now look at the results obtained from the output of the framework as a whole. For the complete complex event definitions we refer to Appendix A. We have statistically determined the alarming threshold to be 0.6. All use cases described can be queried for all the complex events determined in the rules. However, often the probability of a complex event does not cross the alarming threshold once in the video or the complex event refers to persons in the background. Therefore, these complex events are omitted from the graphs and only one or two complex events are displayed depending on the use case.

Person leaving a bag

The first use case inputs a video in which a person enters the frame with a suitcase, labelled as bag, that she leaves behind. She does not come back and the video is ended before the person leaves the frame. As we are interested in `leaving_bag` complex event, that is how we query the program. The query `holding_bag` is added to be able to see the interaction of both complex events as their rules are dependent on each other (see Listing 7.3). Figure 9.7 shows the output of the framework for this use

Figure 9.6: Examples of wrong outputs for the action recognition algorithm. The images have been cropped compared to the original frame.

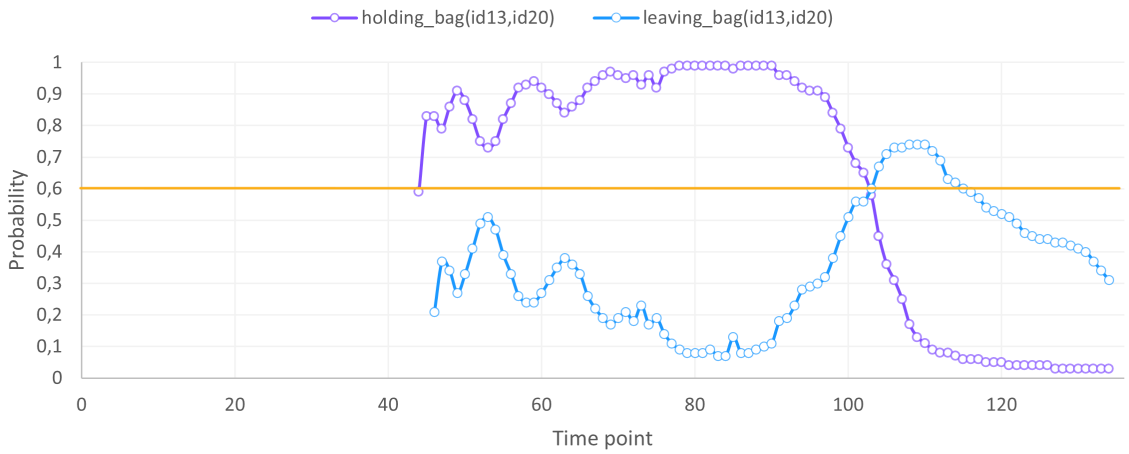


Figure 9.7: Complex event detection for the complex events `leaving_bag` and `holding_bag` for an input video of a person leaving a bag. The yellow line signifies the alarming threshold.

case with the alarming threshold represented by the yellow line. In the first time points, both complex events have not been fired yet which explains the lack of data. Firing a rule refers to all conditions inside the rule being activated such that the rule as a whole is executed. Once a person and a bag have been detected, the complex event rules are fired and the model returns the probability of both complex events. We can clearly see how both complex events are influenced by each other. This is due to the `cached(holdsAt(holding_bag()))` in the initiation of the complex event `leaving_bag` and `initiatedAt(leaving_bag())` in the termination of the complex event `holding_bag`.

We can visualise the frame at which the workforce would be alarmed. This is represented by the first frame after the threshold of 0.6 has been crossed. The resulting frame can be seen in Figure 9.8.

Leaving a bag and returning to it

The second use case and video shows the same scenario except that the person returns to the bag after leaving it. The result can be seen in Figure 9.9. Once again, the probabilities of both complex events follow the events from the video.

Figure 9.10 shows the frame at which the alarm would go off for the complex event `holding_bag` (time point 153) which in this case would represent returning to the bag. The frame in which the bag is left behind is similar to the previous use case.

Figure 9.8: Frame at which the alarm would go off for the complex event `leaving_bag` for an alarming threshold of 0.6.

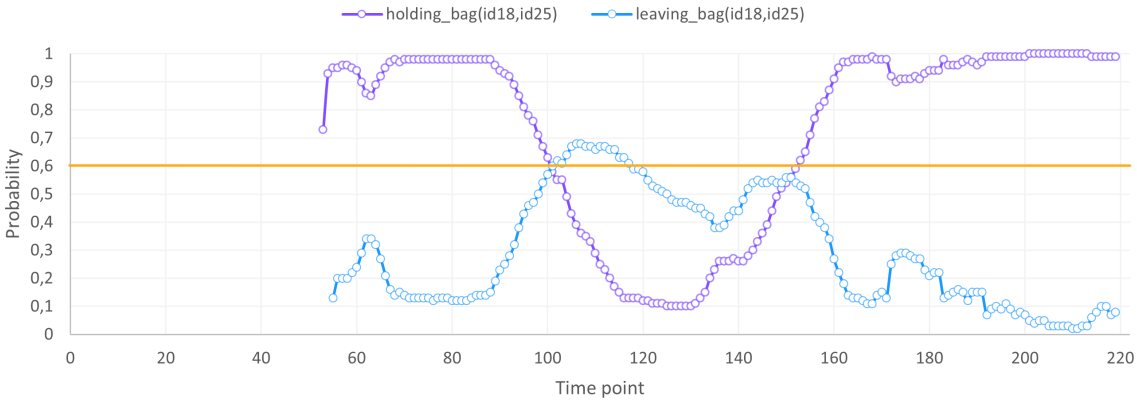


Figure 9.9: Complex event detection for the complex events `leaving_bag` and `holding_bag` for an input video of a person leaving a bag and returning to it.

Figure 9.10: Frame at which the alarm would go off for the complex event `holding_bag` for an alarming threshold of 0.6 after leaving the bag behind.

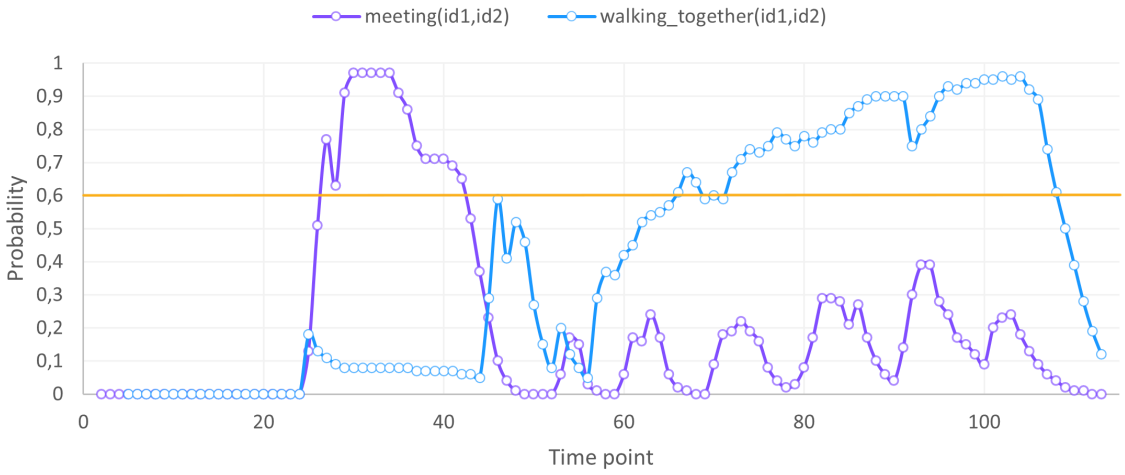


Figure 9.11: Complex event detection for the complex events `meeting` and `walking_together` for an input video of two persons meeting, hugging each other and walking away together.

Figure 9.12: Frame 27 and 66 at which the alarm would go off for the complex events `meeting` (L) and `walking_together` (R) respectively for an alarming threshold of 0.6.

Two people meeting and walking together

The third use case displays two persons walking towards each other, hugging and walking away together. For this video, we are interested in the complex event `meeting` and `walking_together`. The results are displayed in Figure 9.11. The peak for `meeting` can be assigned to the hug the persons give each other. The more spread out peak for `walking_together` can be assigned to the two persons walking away together. The valley in between the peaks is a consequence of the action `walk` not being recognised for both persons. A main drawback of using pixels as a metric for distance becomes clear in this use case. The distance between the persons as they walk away together is relatively constant in world coordinates. In image coordinates, however, the distance between the persons halves. Similarly, when two people are walking away from the camera with a constant real world speed, their speed in pixel will decrease as the distances are smaller far away from the camera. This means the complex event definitions have to be robust for such depth changes and will accept relatively large changes in distance or speed. This can lead to less accurate results either in the timing of the complex event or simply in the existence (or lacking thereof) of the complex event.

Once again we display the frames at which the alarms would go off for both complex events. In this case, we only look at the first crossing. For the complex event `meeting`, this happens at time point 27 and for the complex event `walking_together`, this happens at time point 66. See Figure 9.12.

Movement of one person

The fourth use case looks at simple movements made by one person. For this, three different complex events are used: `stationary`, `abrupt_move` and `stationary_sit`. They are all relatively simple as they are solely based on the average displacement or the action `sit`. The complex event

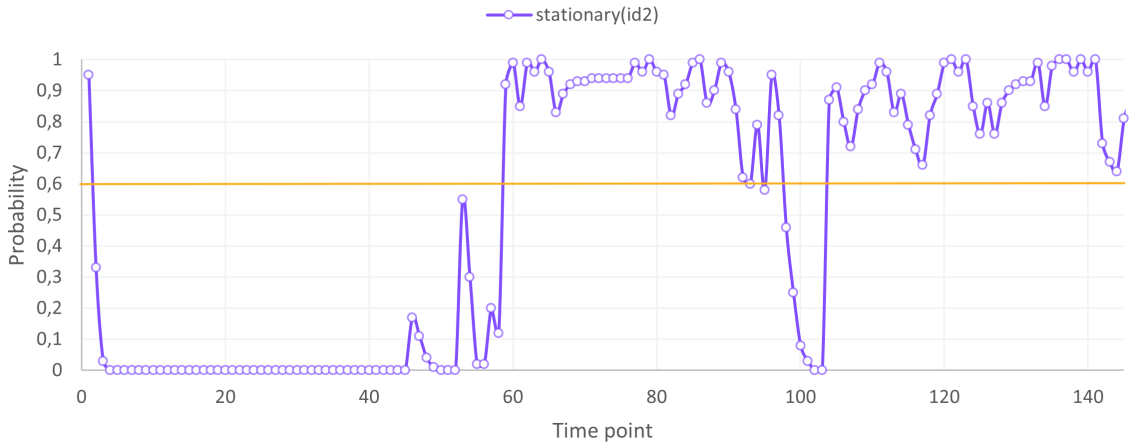


Figure 9.13: Complex event detection for the complex event `stationary` for an input video of a person extensively looking at a house.

`stationary` is fired when the coordinates of a person stay relatively constant over time. The same goes for `stationary_sit` but also includes the action `sit`. The complex event `abrupt_move` is fired in the opposite case, namely, when the coordinates of a person change abruptly. These complex events have been integrated in an elementary way and would not provide much information in a real world scenario. However, the results can give insights into the effect of the granularity of complex event definitions.

We look at four different scenarios:

1. a person walks past a house and stops to watch it. **Complex event:** `stationary`
2. a person walks past a house and stops to take pictures. **Complex event:** `stationary`
3. a person starts walking after which it performs some sudden movements and runs away. **Complex event:** `abrupt_move`
4. a person walks, sits on the ground, stands up and walks on. **Complex event:** `stationary_sit`

The results are displayed in Figures 9.13-9.17 and will be analysed one by one.

In general, the probabilities of the graph in Figure 9.13 correspond to the movement of the person in the first video. The person stops to watch the house around time point 50 until the end of the video. The first time point for the complex event `stationary` always has a high probability. This is due to not having any prior data points which causes an average displacement of zero. We notice the system is very sensitive to noise. A small movement causes a drastic change in probability. Figure 9.14 shows two frames from the video. Frame `a` represents a frame with high probability for the complex event `stationary` and frame `b` represents the valley in the graph around time point 100. The cyclist causes a sudden shift in the coordinates of the person watching the house which explains the low probability at these time points.

The results of the second scenario are displayed in Figure 9.15. Once again we see the peak at the beginning due to the displacement being zero. The period in which the person stands still to take pictures (approximately time point 60-125) is very oscillative. By analysing the images, we can see these oscillations are the consequence of the person extending elbows or arms which in turn causes sudden changes in the dimensions of the bounding box and thus its centre. Ideally, we would like to incorporate more precise object detection such that the mobile phone could be identified as well. This would allow less generic definitions (such as the definition of `stationary`) and thus more specific complex events (such as `gathering_information`).

Scenario 3 looks at the complex event `abrupt_move`. The first oscillations in Figure 9.16 can be assigned to the sudden movements. The increase in probability of the complex event at the end can be assigned to the running motion of the person. We notice that the complex events `abrupt_move` and

(a) Frame 71 (b) Frame 101

Figure 9.14: Frames from the input video of a person walking to a house and stopping to watch it. Frame b has a low probability attached for the complex event `stationary` for the person watching the house.

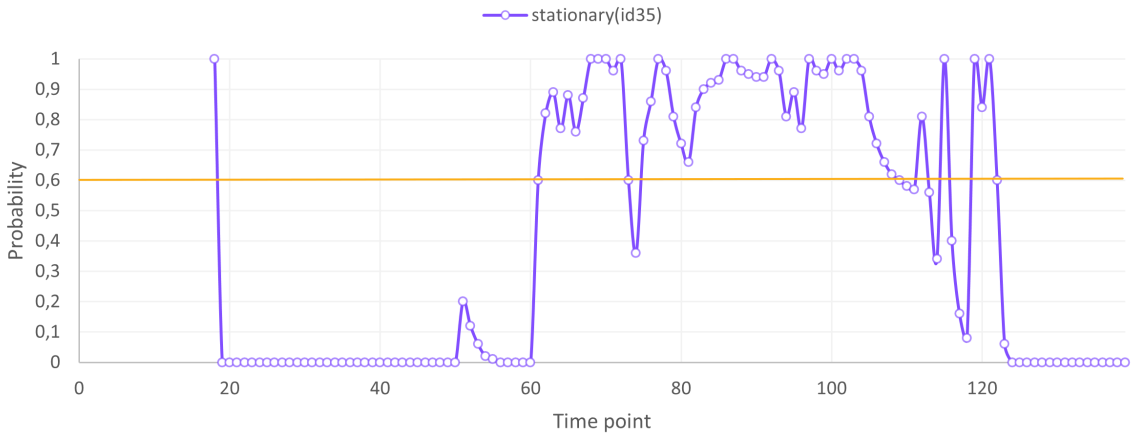


Figure 9.15: Complex event detection for the complex event `stationary` for an input video of a person taking pictures of a house.

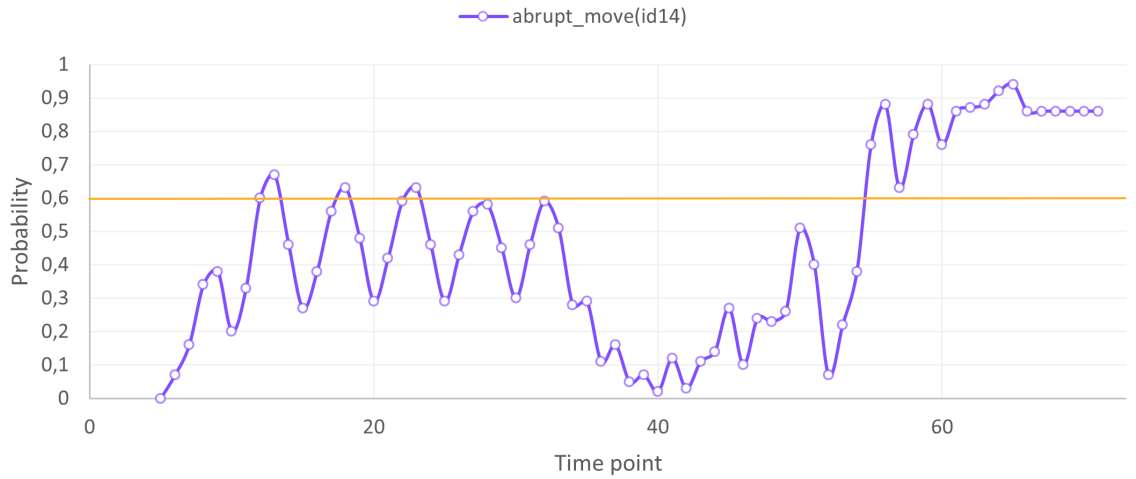


Figure 9.16: Complex event detection for the complex event `abrupt_move` for an input video of a person making sudden moves and running away.

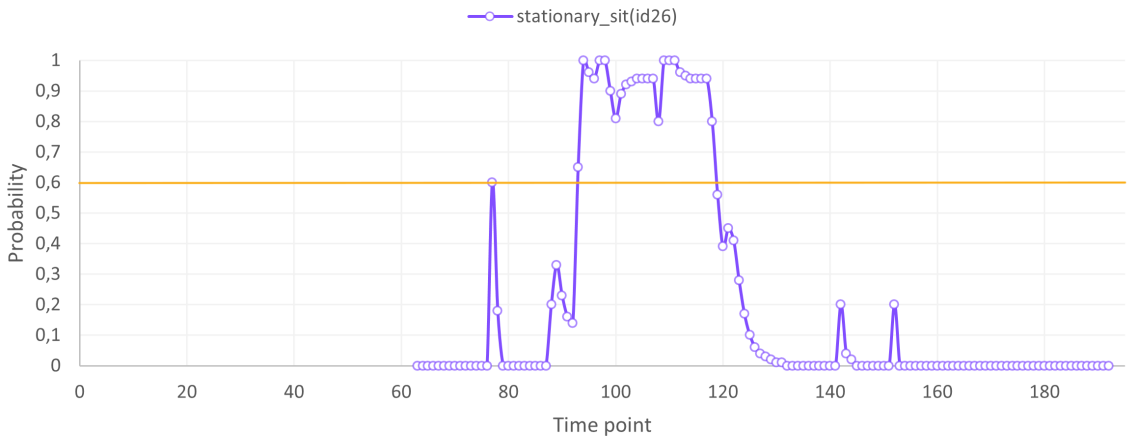


Figure 9.17: Complex event detection for the complex event `stationary_sit` for an input video of a person walking and sitting on the ground.

`stationary` are always fired in videos that entail a person, usually with low probability. This causes many cached complex events compared to a more granular complex event such as `holding_bag` that is only fired if a bag is found in the frame.

In the last scenario, a person sits down around time point 60 and stands up around time point 135. Even though the action recognition already labels the action as sitting, the complex event still has a low probability due to the person still being in motion. Once the person sits still, the complex event attains higher probabilities. Around the time point 120, the action recognition fails to label the sitting person as sitting and the probability decreases. When it does recognise the action as sitting, the person is in motion again which explains the low probabilities.

We conclude these simple complex event definitions are very sensitive to small changes in simple events and cause a lot of unused extra knowledge.

9.3. Profiling

To evaluate the efficiency of the framework, the different processes of the framework are timed for a variety of set-ups. We time the following processes for every time point:

1. Object detection
2. Pose estimation
3. Tracking
4. Action recognition

DeepProbLog also returns the time taken for grounding the program, compiling the AC and evaluating the AC for every query, thus for every complex event. An overview of the processes that are being timed by DeepProbLog can be found in Figure 9.18. Note that in Chapter 8, when speaking about compiling the AC, we included the grounding process. In this section, grounding and compiling are separate processes.

We first look at the relative time taken for the different processes to run. Only one complex event is queried such that the AC is compiled once per time point. The complex event used is `walking_together` with the corresponding video in which two persons meet and walk away together. The total time and average time per frame are displayed in Table 9.1. We can clearly see that the grounding step is relatively time consuming compared to the other processes. The first step low-level processing algorithms (object detection and pose estimation) come second. Both are obtained with YOLOv7. Summing the average time per frame for all processes results in an average time of 0.1877 s per frame. If the feed would come in at a rate of 30 fps, the frame should be processed in less than

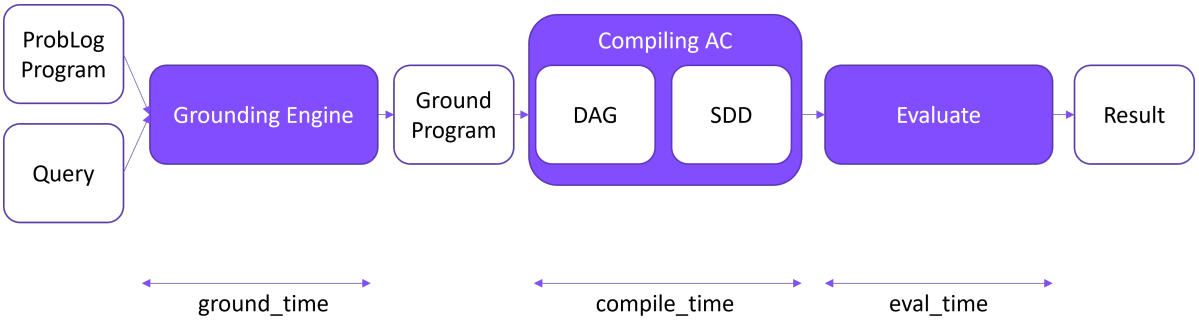


Figure 9.18: Overview of the three processes timed by DeepProbLog. DAG = Directed Acyclic Graph, SDD = Sentential Decision Diagram.

Table 9.1: Total time and average time per frame (in seconds) for each process for the full video in which two persons walk together.

	Detection	Pose	Tracking	Action	Ground	Compile	Evaluate
Total time	3.6346	3.8470	0.2349	0.3515	13.0454	0.2393	0.1570
Average time	0.0323	0.0343	0.0020	0.0027	0.1130	0.0021	0.0013

0.03 s. This means the framework speed should be more than 6 times faster for it to work real-time for a feed with a rate of 30 fps. Note this calculation is based on the main processes of the framework and omits any auxiliary processes. Furthermore, only one complex event is queried.

Let us now look further into the grounding step. The grounding engine takes the program and grounds it for the query. In Chapter 8, we concluded that changing the query was not feasible for this framework. Therefore, we analyse the effect of changing the program. In particular, we look at the effect of the number of detections and the number of cached events as these are added and removed from the program before grounding. The rules themselves are fixed for all tests.

The number of detections is adapted by removing the bicycle class from the object detection, or more specifically from the non-maximum suppression algorithm. This will result in less detections per frame. We compare the results for the initial 7 classes with the results for the classes without the bicycle for the video of a person stealing a bike. This video was chosen due to the high number of detections available. The grounding time per frame is shown in Figure 9.19 for all the frames of the video (with a fps rate of 10) for both cases. The number of detections approximately halves for the case without the bicycle class. Important to note is that none of the rules contains the bicycle predicate which means nothing changes in the number of rules that is fired or not. From the graph, we see a small decrease in grounding time for the lower number of detections. The total grounding time increases with 10% for the case with the bicycle class.

The number of cached events increases for the following reasons: querying for a larger number of complex events, querying for complex events with a lower granularity which are detected more often, caching all the returned complex events instead of only the ones with the highest probability. The first two options are intertwined for this use case. The number of complex events are limited which means querying for more complex events will result in querying for complex events with different granularity. Furthermore, querying for different complex events will also result in different complex events being fired which can also influence the grounding time and is difficult to keep track of. Lastly, the program is grounded for every query at every time point. Querying for more complex events naturally results in a longer grounding time per time point. All this considered, we will only look at the effect of caching only the complex events with highest probability per complex event for every time point. This option is the most independent of the rest of the processes and will, therefore, give the most insight into the effect of the number of cached events on the grounding time. Again, the video of a person stealing a bike is used. We query for five complex events. The results can be seen in Figure 9.20. The grounding time per frame is averaged over all complex events. The grounding time increases for an increasing number of cached events. Although, the increase in grounding time only becomes apparent around 80 cached events. When only caching the results with highest probability, the maximum number of cached events

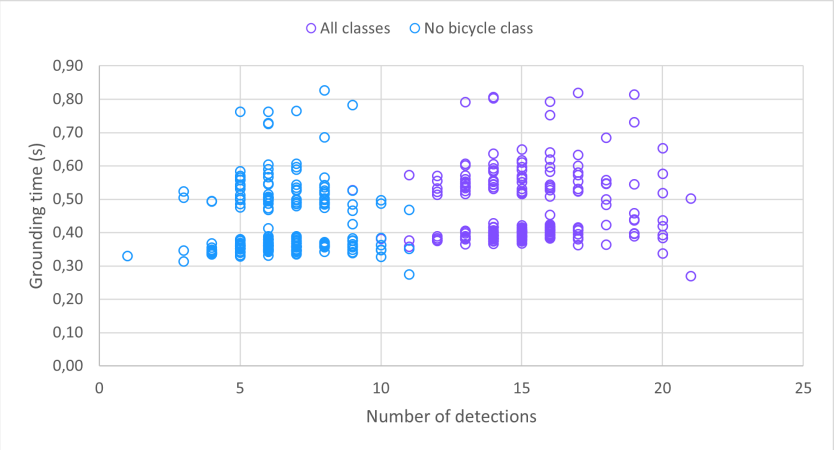


Figure 9.19: Grounding time per frame for different number of detections with and without the bicycle class.

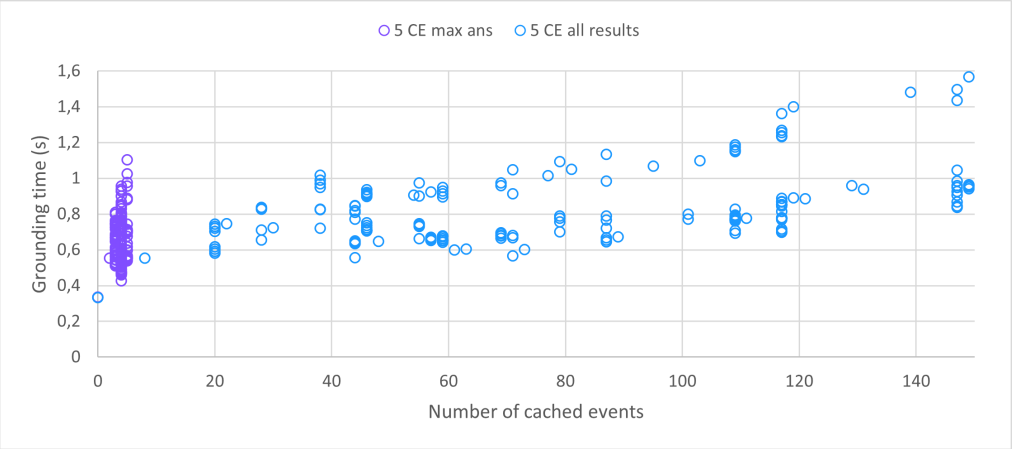


Figure 9.20: Grounding time per frame for different number of cached events for the case of only caching results with the highest probability for every complex event (max ans) or caching all the results.

is equal to the number of queried complex events (5 in this case). On the contrary, when caching all results, the number of cached events increases substantially. This is to be expected. If 5 persons are detected at a time point, there will be 5 different results for every complex event with one argument and $(5-1)*5=20$ different results for every complex event with two arguments. Furthermore, due to the rule of inertia, all cached complex events are returned every time point and thus remain cached, mostly with probability zero.

The length of the video and the frames per second rate also influence the time to obtain all results from a video. However, the framework is designed to work online thus we are mainly interested in the changes that affect the processing time per frame. Increasing resolution, as mentioned earlier, decreases performance thus the resolution is fixed.

This chapter has analysed the framework in three parts. First, the performance of the algorithms that form the low-level processing layer is visually analysed based on the custom dataset. Secondly, the performance of the full framework is assessed on four different use cases with each their own complex events. Lastly, the time performance of the framework is compared for different settings. The framework gives good results on average but is very sensitive to noise. The noise has been limited in the custom dataset which explains the good results. Especially the low level processing layer and the granularity of the complex events should be adapted for real world scenarios.

10 Discussion

We have completed the first two steps of this research as presented in Chapter 1. A proof-of-concept has been designed and we have looked into possible ways to improve inference in DeepProbLog, both for complex even detection on real-world surveillance data. In this chapter, the results presented in Chapter 9 will be discussed as well as the design of the framework as presented in Chapter 6 and touched upon in Chapter 8. This will provide us with enough information to complete step three, namely evaluate the Hellenwaheri framework based on expressivity, efficiency and adaptability.

10.1. Discussion of Experiment Results

The extension of DeepProbLog with Prob-EC for real-world complex event detection provides good results when looking at the output of the framework (the probabilities of the complex events happening) and comparing this to the input video. The timing of the complex event detection is accurate and the range of probabilities obtained from the complex event detection is broad enough to be able to distinguish between a complex event happening or not. As a comparison, the research performed in Vilamala, 2022 uses a probability threshold of 0.3 for a complex event to be detected. In our case, this can be increased to 0.6. The main drawbacks of the current framework are the limited simple events that can be obtained from the frames, the accuracy of the low level processing algorithms and the processing time for one frame. However, without changing the algorithms and the current set-up, the performance of the framework can be improved in multiple ways. From the results, we notice that the probabilities of the complex events oscillate in some cases. This can become a problem if this oscillation happens around the threshold for which the workforce is alarmed. To prevent this, an interval based recognition system on top of the point-based recognition system can be used such as in Mantenoglou et al., 2023. This framework is also an extension of Prob-EC. Furthermore, the cache can be regulated more efficiently. Many cached complex events remain cached with a probability of zero until all frames have been processed. This causes a build up of the cache which increases grounding time.

It is important to realise that the application has greatly been simplified. First, the dataset has purposefully been recorded in such a way that the noise is minimal and the conditions do not complicate the detection of simple events. There aren't any occlusions, the angle is optimal, the lighting is good and main characters in the video do not leave the frame such that tracking is simplified. This ensures relatively high confidence scores for all detections. Secondly, the complex event definitions for this research are limited and (over)fitted to the use case. In a real case scenario, the definitions will be more granular which requires low level processing algorithms with a higher variety in simple events. For example, Caruccio et al., 2019 also looks at facial expression to distinguish between the complex event of leaving luggage or a terrorist attack. This problem is clearly visible when trying to detect either a punching action on a person or two people hugging. Both complex events consist of two people approaching each other and having some form of physical contact while in standing position. A higher granularity will also ensure less firing of the complex event and thus a smaller cache. The complex event `stationary` for example is fired for every person detected. If the definition would be extended to a more specific complex event such as standing still for a longer period of time, the complex event would be fired sporadically. Furthermore, medium-level events should be based on real world coordi-

nates instead of pixels. This will decrease the sensitivity of the framework to small displacements or different conditions. The threshold parameters described in Section 7.5 would not work for a different angle or distance from the camera to the characters in the video. Having more expressive definitions will most likely result in lower probabilities for the overall complex event detections. This relates to the third simplification, the algorithms used to translate the low-level data to simple events are pre-trained off the shelf algorithms. This can, however, also be seen as a complication as training a model on the proper conditions can improve detection accuracy and confidence scores.

As already pointed out in A. Khan et al., 2019, we notice that the performance of the framework is very dependent on the definitions of the complex events. A small change in one of the rules can completely alter the result. Furthermore, the performance of the computer vision algorithms also has a large impact on the output of the entire model. If, for example, a person is not detected or not re-identified, the complex event containing the detection of a person as part of its rules immediately receives a zero probability. This is referred to as pattern uncertainty (Alevizos and Skarlatidis, 2017) and is partially covered by working with probabilistic medium-level events such as `close`. A more complete procedure for pattern uncertainty could improve the detection of complex events. Furthermore, using more advanced low level processing algorithms can prevent this problem in general. For example, we could use Deep SORT [1] to include appearance in the tracking process instead of SORT that only uses data association and state estimation techniques. This means SORT cannot handle occlusions and re-entering of tracked identities. The speed of the low level processing should also be increased significantly as the time taken for low level processing is more than twice the time available for the full processing with a fps rate of 30. Lastly, ensuring the detections and cached events remain minimal without losing information can improve the grounding time. This is, however, insignificant compared to the time that would be gained by not having to compile the AC at every time point for every query. This will be discussed in the next section.

10.2. Discussion of Framework Design

The Hellenwaheri framework has been designed in such a way that the combination of Prob-EC, DeepProbLog and the computer vision layer would function as optimal as possible for complex event detection in a surveillance setting. This means we have created a framework that performs complex event detection within the limitations of the different frameworks it consists of. From previous chapters, we know the design presents one main drawback. That is, the fact that the neural networks used to obtain simple events from the low level data cannot all be integrated as neural predicate. This has multiple consequences. First, DeepProbLog was chosen due to its learning capabilities which are now not being used due to the way the neural networks have been integrated. End-to-end learning is only possible when the loss can be backpropagated through the neural predicate to the weights of the neural network. Secondly, inference efficiency can be improved as has been analysed in this research, but once again many of these improvements would only be possible if the simple events would be obtained through neural predicates.

This makes us wonder what the other possibilities are regarding complex event detection on real-world video data in neurosymbolic AI. Many of the renowned researches on (neural) probabilistic logic programming and statistical relational AI originate from the same research group at KU Leuven. By appraising and referring to each other, these researches have played a prominent role in this research. For example, Raedt et al., 2023 states how neurosymbolic AI should have the logic and learning part as a separate case such that both paradigms can retain their full expressivity. This has led to the criteria that the model should focus on logic which led us to the DeepProbLog framework of which Manhaeve is also the author. However, logic integrated in the neural part of the model has proven to be much faster at inference (Vilamala, 2022). Taking on a broader view, it could be interesting to compare the performance of frameworks that do not focus on logic but integrate it in the neural part of the network with frameworks that do focus on logic for the surveillance application. Other neurosymbolic frameworks that have potential for complex event processing are DeepStochLog, Scallop, NeurASP and DeepSeaProbLog. They use different inference strategies or semantics which potentially makes them more suitable for complex event detection on real-world video data. DeepStochLog, Scallop and NeurASP have been claimed to have better inference processes than DeepProbLog. Their suitability

[1] https://github.com/nwojke/deep_sort

for complex event detection on real-world surveillance data remains to be proven. Especially interesting is DeepSeaProbLog which was published during the course of this research. DeepSeaProbLog is a generalisation of DeepProbLog that allows for continuous probability distributions. This means that we are not limited to using a classification network for the neural predicates which has been a limiting actor in our research. Therefore, more research into DeepSeaProbLog for this application is recommended.

10.3. Evaluation

In this section, the proof-of-concept is evaluated based on expressivity, efficiency and adaptability. This will provide more insight into the usability of the framework and will permit us to answer the research question presented in Chapter 1.

10.3.1. Expressivity

ProbLog is regarded as a very expressive modelling language which has once again been proven in this context. We have not been limited in ways we could define the complex event definitions. However, we have been limited in what simple events could be abstracted from the low-level data. An increase in the variety of information that can be obtained from the video frames provides more knowledge to reason with and thus more expressive definitions. Examples are detecting objects such as a phone and a weapon, facial expressions, accurate body poses or orientation of persons. There is of course the possibility to have a large amount of computer vision networks that each return a simple event but this would be very inefficient. Simple events are returned as probabilistic facts that in turn influence the probabilities of the queried complex events. Obtaining detections with higher confidences results in complex events with higher probabilities. Writing the rules for any use case is a challenging task. The rules need to be efficient such that inference time is minimised. This can be obtained with higher granularity. For example, the complex event `holding_bag` requires a bag which means that as long as no bag is detected, this complex event definition will never be fired. This can prevent unnecessary calculation and clear boundaries between different complex events. However, having many probabilistic facts as part of a complex event definition will result in a lower probability for that complex event as we take the conjunction of these predicates. This is a trade-off that needs to be made. We believe this is a very powerful method to be in control of what should or should not be detected, but is limited by the simple events that are available from the low level data.

Looking back at the criteria from Chapter 3, the framework adheres to all but one. The framework can be used online, focuses on logic, is able to integrate expert knowledge, can reason with spatio-temporal relations and most importantly is probabilistic. However, the framework is not end-to-end differentiable if we consider the computer vision algorithms to be part of the framework. Neural predicates can still be incorporated into the framework and trained using backpropagation but the computer vision algorithms will not be part of this training loop.

10.3.2. Efficiency

As mentioned in Chapter 7, in this section we analyse the time-efficiency of the framework as well as the inference process as a whole. The speed of the framework is unsatisfactory. When using a single complex event, the time taken to process a frame and return the probability of the complex event happening should be more than 6 times faster to run real-time with a fps rate of 30. Querying for more complex events, which is required for real-world scenarios, will only slow down the process even more. The processing of low-level data takes 0.07 s per frame. With an incoming feed at 30 fps, the whole processing pipeline should be finished in under 0.03 s. Thus, both the inference speed and the low-level processing speed should be highly increased for the framework to be used real-time.

In the Hellenwaheri framework, the AC has to be recompiled for every complex event at every time point. We know that compiling the AC is the most costly process in DeepProbLog inference. There are three reasons for having to recompile the AC at every time point: the queries, the cached complex events and the simple events. The query contains a time point which prevents us from re-using the same query (and thus the same AC). The cached complex events are grounded for the different tracking IDs identified in the videos which means they are unpredictable. The simple events are not obtained through the neural predicates but added to the knowledge base before compiling. Chapter 8 discusses ways to reduce the need for compiling the AC. However, these cannot be implemented for our case as is. Assuming all the simple events would be obtained though the neural predicates, the time point

is transformed to a tensor variable and the cached grounded queries are not added to the knowledge base before compiling, using the AC cache from DeepProbLog could improve efficiency. We wonder if these improvements are worth the effort as DeepProbLog has been accused of having scalability issues when the complexity increases Apriceno et al., 2021.

10.3.3. Adaptability

The framework is adaptable in the sense that it does not have to be retrained when adding complex event definitions. This provides the ability to adapt the framework for altering conditions while in operation. If properly written, rules and definitions can be used for multiple applications by simply adapting the domain knowledge parameters.

We now look at how the proof-of-concept should be adapted to be operational. Evidently, the knowledge of the workforce, the experts in this case, will need to be translated to complex and medium-level event definitions. The knowledge should be translated in such a way that it is correct and does not contain bias from the workforce. The program will be a large database of complex event definitions, medium-level event definitions and simple event facts that are fired hierarchically through the different predicates. A next step consists of choosing or training the networks for the translation of video data to simple events needed for the complex event definitions. Multiple adaptations to the framework could improve the accuracy of this step. For example, by introducing a depth camera or using background subtraction in the case of static cameras. As mentioned earlier, the framework is too slow to work real-time thus the speed of the framework should drastically be improved. The program will be queried for many complex events at once at every time point and will thus return the probability of every complex event at every time point.

An important aspect of every design is the interaction with the user. The framework should complement the work of the workforce and not completely replace it. It should ensure the workforce does not get bored and understands the decision process of the framework such that follow-up decisions can be made swiftly. Ideally, the framework returns its line of reasoning in a human understandable form. For example, by highlighting entities that have contributed to a raised alarm in the video feed.

10.4. Answer to the Research Question

We have now completed all three steps as introduced in Chapter 1 and are therefore able to answer the research question:

”To what extent is a neural probabilistic logic programming framework suitable for online complex event detection on real-world surveillance video data while efficiently making use of available expert knowledge?”

DeepProbLog itself adheres to all criteria presented in Chapter 3, however, the Hellenwaheri framework does not anymore. The framework is not end-to-end differentiable and thus not trainable due to the low-level detection layer not being integrated as neural predicates. This makes the inference process even more inefficient than it already was and causes the framework to be excessively slow. A scenario where all simple events are integrated through neural predicates is not realistic at this point for nPLP frameworks due to the limitation on the type of neural network. As a consequence, nPLP frameworks are too limited for complex event detection on real-world surveillance data. However, DeepProbLog still remains suitable for complex event detection on more simple scenarios such as audio data (Preece et al., 2021) or synthetic data (Apriceno et al., 2021). Furthermore, the use of knowledge offers great advantages such as human understandable symbols and more control over the decisions made by the framework.

11

Conclusion

Due to a rise in organised crime in the Netherlands, the Marechaussee needs to deploy its workforce more efficiently. To this end, the Marechaussee has shown interest in using an intelligent video surveillance system for residential security. This research explores the use of neurosymbolic AI, specifically a neural probabilistic logic programming (nPLP) framework, for online complex event detection in real-world surveillance video data. Complex events refer to situations that involve a spatio-temporal combination of multiple atomic events. The framework should follow six criteria: (1) Integration of expert knowledge to reduce complexity and enhance training efficiency; (2) Incorporation of probabilistic methods to handle uncertainties in real-world complex events; (3) Online processing capability for immediate recognition of complex events in the video stream; (4) Ability to reason with spatio-temporal data to analyse temporal events and capture interactions; (5) Focusing on logic instead of neural networks to retain full expressivity of the logic; and (6) End-to-end differentiability for simplified training.

DeepProbLog, a well documented and maintained nPLP framework, is theoretically able to follow all six criteria but has been accused of having a low inference speed and has limited temporal reasoning capabilities. Prob-EC, a probabilistic version of the Event Calculus, is combined with DeepProbLog to enhance the latter. This forms the Hellenwaheri framework. A proof-of-concept is designed to test the framework on expressivity, efficiency and adaptability and assess the suitability of this type of framework for the use case.

The results are as follows:

Expressivity As ProbLog is a highly expressive programming language, so is DeepProbLog. This expressivity is enhanced by integrating Prob-EC. The expressivity of the framework is limited by the variety and accuracy of the simple events obtained from the video data. The output of the framework is highly dependent on the simple events generated and their accuracy. Choosing efficient and suitable algorithms for the computer vision layer is a challenging task due to the noise that comes with real world data.

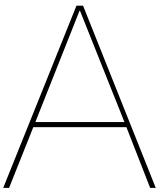
Efficiency The low level processing layer and cache have been integrated as part of the ProbLog program which is highly inefficient. Together with the queries that change for every time point, these processes force the AC to be recompiled at every time point for every complex event which is the most time consuming process of the DeepProbLog framework. Furthermore, the low level processing algorithms are slow and are not part of the DeepProbLog training loop which causes the framework to fail for criteria 6. To address these issues, strategies are explored for reducing the compiling frequency or decreasing the compiling time. The integration of the low level processing neural networks as neural predicates could provide end-to-end learning and improve inference efficiency. However, a scenario where all simple events are integrated through neural predicates is not realistic at this point for nPLP frameworks due to the limitation on the type of neural network.

Adaptability Focusing on logic has the advantage of allowing the addition of complex event definitions without retraining. This increases the adaptability. To operationalise the framework, correct and

unbiased rules need to be crafted by experts which can be challenging. Low level processing algorithms need to be carefully selected as these have a large impact on the performance of the system. The time complexity of the inference, however, remains a bottleneck in deploying the framework and does not come with a quick fix.

We conclude an nPLP framework is not suitable for complex event detection on real-world surveillance data at this point in time due to the current constraints of ProbLog and DeepProbLog. Alongside the shortcomings, the framework has shown the great advantages of using knowledge for this use case. The use of human-understandable symbols provides more explainability and defining and maintaining rules provides more control over the decisions made by the framework. Neurosymbolic AI should therefore not be written off, on the contrary, let this research be a foundation and inspiration for using neurosymbolic AI for complex event detection on real-world data.

We end this thesis with a quote by none other than Isaac Asimov: *"The most exciting phrase to hear in science, the one that heralds the most discoveries, is not 'Eureka!' but 'That's funny…'"*



Event Definitions

```
% Asserts and retracts dynamic predicates which do not appear
% in probabilistic facts or rules at the starting program
% to avoid unknown predicate errors

cached(aux).
happensAt(aux1,aux2).


% ===== DEFINE STATICALLY DETERMINED FLUENTS

Prob::holdsAtMacro(close(Person1, Person2, D) = true, T) :-
   calculateDistance(Person1, Person2, T, Dist),
   Prob is max(0,(1-max(0,(Dist-100))/D)). % Dist =< Threshold.

Prob::holdsAtMacro(close(Person1, Person2, D) = false, T) :-
   calculateDistance(Person1, Person2, T, Dist),
   Prob is min(1,(max(0,(Dist-100))/D)). % Dist > Threshold.

Prob::holdsAtMacro(moving_closer(Person1, Person2) = true, T):-
   closingDist(D),
   prevTimepoint(T, Tprev),
   calculateDistance(Person1, Person2, Tprev, Dist1),
   calculateDistance(Person1, Person2, T, Dist2),
   DeltaDist is max(Dist1-Dist2,0),
   Prob is min(1,DeltaDist/D).

Prob :: holdsAtMacro(same_speed(Person1,Person2)=true, T):-
   % writeln('Checkpoint 4'),
   holdsAtIE(avg_displ(Person1)=Speed1,T),
   holdsAtIE(avg_displ(Person2)=Speed2,T),
   SpeedDif is abs(Speed1-Speed2),
   maxSpeedDif(D),
   Prob is max(0,(1-SpeedDif/D)).

Prob::holdsAtMacro(abrupt_displ(Displ)=true, T):-
   abruptDispl(D),
   Prob is min(1,(Displ/D)).

Prob::holdsAtMacro(abrupt_displ(Displ)=false, T):-
   abruptDispl(D),
```

```
41    Prob is max(0,(1-Displ/D)).
42
43 Prob::holdsAtMacro(get_prob_station(Displ)=true, T):-
44    stationDispl(D),
45    Prob is max(0,1-Displ/D).
46
47 Prob::holdsAtMacro(get_prob_station(Displ)=false, T):-
48    movingDispl(D),
49    Prob is min(1,Displ/D).
50
51 calculateDistance(Person1, Person2, T, Dist):-
52    \+ Person1 = Person2,
53    % writeln('Checkpoint 1'),
54    holdsAtIE(coord(Person1) = (X1, Y1), T ),
55    holdsAtIE(coord(Person2) = (X2, Y2), T ),
56    XDiff is abs(X1-X2),
57    YDiff is abs(Y1-Y2),
58    SideA is XDiff*XDiff,
59    SideB is YDiff*YDiff,
60    Temp is SideA + SideB,
61    Dist is sqrt(Temp).
62 % ================================================
63 % COMPLEX EVENT: holding_bag(Person, Object)
64 % ================================================
65 initiatedAt(holding_bag(Person,Bag) = true, T):-
66    happensAt(person(Person), T),
67    happensAt(bag(Bag), T),
68    holdDist(D),
69    holdsAtMacro(close(Person, Bag, D)=true, T).
70
71 initiatedAt(holding_bag(Person,Bag) = false, T):-
72    happensAt(person(Person), T),
73    happensAt(bag(Bag), T),
74    initiatedAt(leaving_bag(Person, Bag) = true, T).
75
76 % ================================================
77 % COMPLEX EVENT: leaving_bag(Person, Object)
78 % ================================================
79 initiatedAt(leaving_bag(Person, Bag) = true, T):-
80    happensAt(person(Person), T),
81    happensAt(bag(Bag), T),
82    prevTimepoint(T, Tprev),
83    cached(holdsAt(holding_bag(Person, Bag) = true, Tprev)),
84    holdDist(D),
85    holdsAtMacro(close(Person, Bag, D)=false, T).
86
87 initiatedAt(leaving_bag(Person, Bag) = false, T):-
88    happensAt(bag(Bag), T),
89    \+ happensAt(person(Person), T).
90
91 initiatedAt(leaving_bag(Person, Bag) = false, T):-
92    happensAt(bag(Bag), T),
93    happensAt(person(Person), T),
94    holdDist(D),
95    holdsAtMacro(close(Person, Bag, D)=true, T).
96
```

```
% ================================================
% COMPLEX EVENT: meeting(Person1, Person2)
% ================================================
initiatedAt(meeting(Person1, Person2) = true, T):-
    happensAt(person(Person1), T),
    happensAt(person(Person2), T),
    \+ Person1 = Person2,
    holdsAtMacro(moving_closer(Person1, Person2) = true, T),
    meetingDist(D),
    holdsAtMacro(close(Person1, Person2, D)=true, T).

initiatedAt(meeting(Person1,Person2) = false, T):-
    happensAt(person(Person1), T),
    happensAt(person(Person2), T),
    meetingDist(D),
    holdsAtMacro(close(Person1, Person2, D)=false, T).


% ================================================
% COMPLEX EVENT: walking_together(Person1, Person2)
% ================================================
initiatedAt(walking_together(Person1, Person2) = true, T):-
    happensAt(person(Person1), T),
    happensAt(person(Person2), T),
    \+ Person1 = Person2,
    happensAt(walk(Person1), T),
    happensAt(walk(Person2), T),
    walkDist(D),
    holdsAtMacro(close(Person1, Person2, D)=true, T),
    prevTimepoint(T,Tprev),
    holdsAtMacro(close(Person1, Person2, D)=true, Tprev),
    holdsAtMacro(same_speed(Person1,Person2)=true,T).


initiatedAt(walking_together(Person1,Person2) = false, T):-
    happensAt(person(Person1), T),
    happensAt(person(Person2), T),
    walkDist(D),
    holdsAtMacro(close(Person1, Person2, D)=false, T).

% ================================================
% COMPLEX EVENT: abrupt_move(Person)
% ================================================
initiatedAt(abrupt_move(Person) = true, T):-
    happensAt(person(Person), T),
    holdsAtIE(avg_displ(Person)=Displ,T),
    holdsAtMacro(abrupt_displ(Displ)=true, T).

initiatedAt(abrupt_move(Person) = false, T):-
    happensAt(person(Person), T),
    holdsAtIE(avg_displ(Person)=Displ,T),
    holdsAtMacro(abrupt_displ(Displ)=false, T).

% ================================================
% COMPLEX EVENT: stationary(Person)
% ================================================
```

```
153
154 initiatedAt(stationary(Person) = true, T):-
155    happensAt(person(Person), T),
156    holdsAtIE(avg_displ(Person)=Displ,T),
157    holdsAtMacro(get_prob_station(Displ) = true, T).
158
159 initiatedAt(stationary(Person) = false, T):-
160    happensAt(person(Person), T),
161    holdsAtIE(avg_displ(Person)=Displ,T),
162    holdsAtMacro(get_prob_station(Displ) = false, T).
163
164 % ================================================
165 % COMPLEX EVENT: stationary_sit(Person)
166 % ================================================
167
168 initiatedAt(stationary_sit(Person) = true, T):-
169    happensAt(person(Person), T),
170    happensAt(sit(Person),T),
171    holdsAtIE(avg_displ(Person)=Displ,T),
172    holdsAtMacro(get_prob_station(Displ) = true, T).
173
174 initiatedAt(stationary_sit(Person) = false, T):-
175    happensAt(person(Person), T),
176    happensAt(jump(Person),T).
177
178 initiatedAt(stationary_sit(Person) = false, T):-
179    happensAt(person(Person), T),
180    holdsAtIE(avg_displ(Person)=Displ,T),
181    holdsAtMacro(get_prob_station(Displ) = false, T).
```

Listing A.1: Complex and medium-level event definitions

Bibliography

Ahmad, K., & Conci, N. (2019). How deep features have improved event recognition in multimedia: A survey. *ACM Trans. Multimedia Comput. Commun. Appl*, *15*. https://doi.org/10.1145/3306240

Alevizos, E., & Skarlatidis, A. (2017). Probabilistic complex event recognition: A survey. *Surv*, *50*, 31. https://doi.org/10.1145/3117809

Apriceno, G., Kessler, F. B., & Passerini, I. A. (2021). A neuro-symbolic approach to structured event recognition. https://doi.org/10.4230/LIPIcs.TIME.2021.11

Apriceno, G., Passerini, A., & Serafini, L. (2022). A Neuro-Symbolic Approach for Real-World Event Recognition from Weak Supervision. *DROPS-IDN/17259*, *247*. https://doi.org/10.4230/LIPICS.TIME.2022.12

Baxter, R. H., Robertson, N. M., & Lane, D. M. (2015). Human behaviour recognition in data-scarce domains. *Pattern Recognition*, *48*, 2377–2393. https://doi.org/10.1016/J.PATCOG.2015.02.019

Ben Mabrouk, A., & Zagrouba, E. (2018). Abnormal behavior recognition for intelligent video surveillance systems: A review. *Expert Systems with Applications*, *91*, 480–491. https://doi.org/10.1016/J.ESWA.2017.09.029

Bewley, A., Ge, Z., Ott, L., Ramos, F., & Upcroft, B. (2016). Simple online and realtime tracking. *CoRR*, *abs/1602.00763*. http://arxiv.org/abs/1602.00763

Bouneffouf, D., & Aggarwal, C. C. (2022). Survey on Applications of Neurosymbolic Artificial Intelligence. https://arxiv.org/abs/2209.12618v1

Caruccio, L., Polese, G., Tortora, G., & Iannone, D. (2019). EDCAR: A knowledge representation framework to enhance automatic video surveillance. *Expert Systems With Applications*, *131*, 190–207. https://doi.org/10.1016/j.eswa.2019.04.031

Chen, F. (2021). GitHub - felixchenfy/Realtime-Action-Recognition: Apply ML to the skeletons from OpenPose; 9 actions; multiple people. https://github.com/felixchenfy/Realtime-Action-Recognition?tab=readme-ov-file

De Raedt, L., Kimmig, A., & Toivonen, H. (2007). Problog: A probabilistic prolog and its application in link discovery. *IJCAI 2007, Proceedings of the 20th international joint conference on artificial intelligence*, 2462–2467.

Fierens, D., den Broeck, G. V., Renkens, J., Shterionov, D. S., Gutmann, B., Thon, I., Janssens, G., & Raedt, L. D. (2013). Inference and learning in probabilistic logic programs using weighted boolean formulas. *CoRR*, *abs/1304.6810*. http://arxiv.org/abs/1304.6810

Garcez, A. d., & Lamb, L. C. (2023). Neurosymbolic AI: the 3rd wave. *Artificial Intelligence Review*. https://doi.org/10.1007/s10462-023-10448-w

Hitzler, P., Sarker, K., & Eberhart, A. (Eds.). (2023). *Neuro-Symbolic Artificial Intelligence: the State of the Art* (Vol. 342). IOS Press.

Karbalaie, A., Abtahi, F., & Sjöström, M. (2022). Event detection in surveillance videos: A review. *Multimedia Tools and Applications*, *81*, 35463–35501. https://doi.org/10.1007/s11042-021-11864-2

Kardas, K., & Cicekli, N. K. (2017). SVAS: Surveillance Video Analysis System. *Expert Systems With Applications*, *89*, 343–361. https://doi.org/10.1016/j.eswa.2017.07.051

Kastrati, M., & Biba, M. (2019). Statistical relational learning: A state-of-the-art review. *Journal of Engineering Technology and Applied Sciences*. https://doi.org/10.30931/jetas.594586

Khan, A., Bozzato, L., Serafini, L., Lazzerini, B., et al. (2019). Visual reasoning on complex events in soccer videos using answer set programming. *GCAI 2019. Proceedings of the 5th Global Conference on Artificial Intelligence*, *65*, 42–53.

Khan, J., Curry, E., & Khan, M. J. (2020). Neuro-symbolic Visual Reasoning for Multimedia Event Processing: Overview, Prospects and Challenges. http://ceur-ws.org

Kimmig, A., Demoen, B., De Raedt, L., Costa, V. S., & Rocha, R. (2011). On the implementation of the probabilistic logic programming language problog. *Theory and Practice of Logic Programming*, *11*(2-3), 235–262. https://doi.org/10.1017/S1471068410000566

Kowalski, R., & Sergot, M. (1985). A logic-based calculus of events. *New Generation Computing*, *4*, 67–95. https://doi.org/10.1007/BF03037383

Latapie, H., Kilic, O., Thórisson, K. R., Wang, P., & Hammer, P. (2022). Neurosymbolic Systems of Perception and Cognition: The Role of Attention. *Frontiers in Psychology*, *13*(May), 1–10. https://doi.org/10.3389/fpsyg.2022.806397

Li, Z., & Huang, J. (2023). Scallop: A language for neurosymbolic programming; scallop: A language for neurosymbolic programming. https://doi.org/10.1145/3591280

Manhaeve, R., Dumančić, S., Kimmig, A., Demeester, T., & De Raedt, L. (2018). DeepProbLog: Neural Probabilistic Logic Programming. https://bitbucket.org/problog/deepproblog.

Manhaeve, R., Dumančić, S., Kimmig, A., Demeester, T., & De Raedt, L. (2021). Neural probabilistic logic programming in deepproblog. *Artificial Intelligence*, *298*, 103504. https://doi.org/https://doi.org/10.1016/j.artint.2021.103504

Manhaeve, R., Marra, G., & De Raedt, L. (2021). Approximate Inference for Neural Probabilistic Logic Programming. *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning*, 475–486. https://doi.org/10.24963/kr.2021/45

Mantenoglou, P., Artikis, A., & Paliouras, G. (2023). Online event recognition over noisy data streams. *International Journal of Approximate Reasoning*, 108993. https://doi.org/10.1016/j.ijar.2023.108993

Mao, J., Gan, C., Kohli, P., Tenenbaum, J. B., & Wu, J. (2019). The Neuro-Symbolic Concept Learner: Interpreting Scenes, Words, and Sentences From Natural Supervision. *7th International Conference on Learning Representations, ICLR 2019*. https://arxiv.org/abs/1904.12584v1

Marcus, G. (2020). The next decade in ai: Four steps towards robust artificial intelligence.

Marra, G., Dumančić, S., Manhaeve, R., & Raedt, L. D. (2021). From statistical relational to neural symbolic artificial intelligence: A survey. *CoRR*, *abs/2108.11451*. https://arxiv.org/abs/2108.11451

Ministerie van Justitie en Veiligheid. (2022). *Kamerbrief voortgang versterking stelsel bewaken en beveiligen*.

Nawaratne, R., Alahakoon, D., De Silva, D., & Yu, X. (2020). Spatiotemporal anomaly detection using deep learning for real-time video surveillance. *IEEE Transactions on Industrial Informatics*, *16*(1), 393–402. https://doi.org/10.1109/TII.2019.2938527

Önal, I., Kardaş, K., Rezaeitabar, Y., Bayram, U., Bal, M., Ulusoy, I., & Çiçekli, N. K. (2013). A framework for detecting complex events in surveillance videos. *Electronic Proceedings of the 2013 IEEE International Conference on Multimedia and Expo Workshops, ICMEW 2013*. https://doi.org/10.1109/ICMEW.2013.6618411

Onderzoeksraad voor Veiligheid. (2023). *Bewaken en beveiligen: Lessen uit drie beveiligingssituaties*. www.onderzoeksraad.nl.

Persia, F., Bettini, F., & Helmer, S. (2017a). An Interactive Framework for Video Surveillance Event Detection and Modeling.

Persia, F., Bettini, F., & Helmer, S. (2017b). Labeling the frames of a video stream with interval events. *Proceedings - IEEE 11th International Conference on Semantic Computing, ICSC 2017*, 204–211. https://doi.org/10.1109/ICSC.2017.55

Persia, F., D'Auria, D., & Pilato, G. (2020). An overview of video surveillance approaches. *Proceedings - 14th IEEE International Conference on Semantic Computing, ICSC 2020*, 287–294. https://doi.org/10.1109/ICSC.2020.00058

Preece, A. D., Braines, D., Cerutti, F., Furby, J., Hiley, L., Kaplan, L., Law, M., Russo, A., Srivastava, M., Roig Vilamala, M., & Xing, T. (2021). Coalition situational understanding via explainable neuro-symbolic reasoning and learning, 61. https://doi.org/10.1117/12.2587850

Qiu, J., Wang, L., Wang, Y., & Hu, Y. H. (2020). Multi-event modeling and recognition using extended petri nets. *IEEE Access*, *8*, 37879–37890. https://doi.org/10.1109/ACCESS.2020.2975095

Raedt, L. D., Dumančić, S., Manhaeve, R., & Marra, G. (2020). From statistical relational to neuro-symbolic artificial intelligence.

Raedt, L. D., & Kimmig, A. (2015). Probabilistic (logic) programming concepts. *Machine Learning*, *100*, 5–47. https://doi.org/10.1007/s10994-015-5494-z

Raedt, L. D., Manhaeve, R., Dumančić, S., Demeester, T., Kimmig, A., & Leuven, K. U. (2023). *Neuro-symbolic = neural + logical + probabilistic*.

Renkens, J., Broeck, G. V. D., & Nijssen, S. (2012). K-optimal: A novel approximate inference algorithm for problog. *Machine Learning*, *89*, 215–231. https://doi.org/10.1007/s10994-012-5304-9

Riguzzi, F., & Swift, T. (2018). A survey of probabilistic logic programming. In *Declarative logic programming: Theory, systems, and applications* (pp. 185–228). Association for Computing Machinery; Morgan & Claypool. https://doi-org.tudelft.idm.oclc.org/10.1145/3191315.3191319

Shidik, G. F., Noersasongko, E., Nugraha, A., Andono, P. N., Jumanto, J., & Kusuma, E. J. (2019). A systematic review of intelligence video surveillance: Trends, techniques, frameworks, and datasets. *IEEE Access*, *7*, 170457–170473. https://doi.org/10.1109/ACCESS.2019.2955387

Skarlatidis, A., Artikis, A., Filippou, J., & Paliouras, G. (2015). A probabilistic logic programming event calculus. *TLP*, *15*, 213–245. https://doi.org/10.1017/S1471068413000690

Skarlatidis, A., & Vouros, G. G. (2014). *Event recognition under uncertainty and incomplete data*.

Smet, L. D., Manhaeve, R., Marra, G., & Martires, P. Z. D. (2022). Tensorised probabilistic inference for neural probabilistic logic programming. *The 5th Workshop on Tractable Probabilistic Modeling*. https://openreview.net/forum?id=6Kpbq2Y2IK6

Sreelekha, S. (2018). NeuroSymbolic integration with uncertainty. *Annals of Mathematics and Artificial Intelligence*, *84*(3-4), 201–220. https://doi.org/10.1007/S10472-018-9605-Y/METRICS

Suchan, J., Bhatt, M., & Varadarajan, S. (2021). Commonsense visual sensemaking for autonomous driving – On generalised neurosymbolic online abduction integrating vision and semantics. *Artificial Intelligence*, *299*, 103522. https://doi.org/10.1016/J.ARTINT.2021.103522

Susskind, Z., Arden, B., John, L. K., Stockton, P., & John, E. B. (2021). Neuro-symbolic ai: An emerging class of ai workloads and their characterization. *arXiv preprint arXiv:2109.06133*.

Taisuke, S. (1995). A statistical learning method for logic programs with distribution semantics. *Proceedings of the 12th international conference on logic programming (ICLP'95)*, 715–729.

Vilamala, M. R. (2022). Combining neural networks with symbolic approaches to perform complex event processing on non-symbolic data.

Vilamala, M. R., Taylor, H., Xing, T., Garcia, L., Srivastava, M., Kaplan, L., Preece, A., Kimmig, A., & Cerutti, F. (2020). A hybrid neuro-symbolic approach for complex event processing *. https://github.com/MarcRoigVilamala/DeepProbCEP

Vilamala, M. R., Xing, T., Taylor, H., Garcia, L., Srivastava, M., Kaplan, L., Preece, A., Kimmig, A., & Cerutti, F. (2023). Deepprobcep: A neuro-symbolic approach for complex event processing in adversarial settings. *Expert Systems with Applications*, *215*, 119376. https://doi.org/10.1016/J.ESWA.2022.119376

Wang, C.-Y., Bochkovskiy, A., & Liao, H.-Y. M. (2023). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Winters, T., Marra, G., Manhaeve, R., & Raedt, L. D. (2022). *Deepstochlog: Neural stochastic logic programming*. www.aaai.org

Xing, T., Garcia, L., Vilamala, M. R., Cerutti, F., Kaplan, L., Preece, A., & Srivastava, M. (2020). Neuroplex: Learning to detect complex events in sensor networks through knowledge injection. *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*, 489–502. https://doi.org/10.1145/3384419.3431158