A Neuro-Symbolic Approach to Structured Event Recognition

Gianluca Apriceno
University of Trento, Italy
Fondazione Bruno Kessler, Italy

Andrea Passerini
University of Trento, Italy

Luciano Serafini
Fondazione Bruno Kessler, Italy

Abstract

Events are structured entities with multiple components: the event type, the participants with their roles, the outcome, the sub-events etc. A fully end-to-end approach for event recognition from raw data sequence, therefore, should also solve a number of simpler tasks like recognizing the objects involved in the events and their roles, the outcome of the events as well as the sub-events. Ontological knowledge about event structure, specified in logic languages, could be very useful to solve the aforementioned challenges. However, the majority of successful approaches in event recognition from raw data are based on purely neural approaches (mainly recurrent neural networks), with limited, if any, support for background knowledge. These approaches typically require large training sets with detailed annotations at the different levels in which recognition can be decomposed (e.g., video annotated with object bounding boxes, object roles, events and sub-events). In this paper, we propose a neuro-symbolic approach for structured event recognition from raw data that uses "shallow" annotation on the high-level events and exploits background knowledge to propagate this supervision to simpler tasks such as object classification. We develop a prototype of the approach and compare it with a purely neural solution based on recurrent neural networks, showing the higher capability of solving both the event recognition task and the simpler task of object classification, as well as the ability to generalize to events with unseen outcomes.

2012 ACM Subject Classification Computing methodologies → Temporal reasoning; Computing methodologies → Activity recognition and understanding

Keywords and phrases Event recognition, learning and reasoning, neuro-symbolic integration

Digital Object Identifier 10.4230/LIPIcs.TIME.2021.11

Funding This research was partially supported by TAILOR, a project funded by EU Horizon 2020 research and innovation programme under GA No 952215.

Acknowledgements The open access publication of this article was supported by the Alpen-Adria-Universität Klagenfurt, Austria.

1 Introduction

Events are structured entities with multiple components and relations with other entities [18]. The most important components of an event are the event type, the participants with their roles, the sub-events, and the event outcome. Therefore, the approaches for full fledged event recognition should be able to extract the information about all the components of the events that happen in a data sequence. To this aim, a system for event detection should solve a number of different simpler tasks like recognizing the objects involved in the events and their roles, the outcome of the events as well as the sub-events. In this context, having background knowledge about the event structure, specified in logic languages, could be very useful to solve the aforementioned challenges. However, looking at [22], one can see that the majority of neural approaches (also knows as sub-symbolic) applied in event recognition

strictly rely on the features learnt by the underlying networks with limited, if any, support for background knowledge. Furthermore, the training of the underlying networks of these approaches requires a large amount of training data with a detailed supervision on all the events' components (e.g., a video annotated with events, sub-events, object roles and object bounding boxes). Alternatively, one could think to have data with an annotation limited to the occurrence of an event (i.e., a "shallow" annotation) and exploit the background knowledge to infer information on the event components. For example, if a video clip is annotated with the event "John is preparing a cappuccino to Mary", one can infer from the background knowledge that the video is showing at least two people, one male and one female, that John is preparing the cappuccino by mixing milk and coffee into two cups, etc. All of these inferred facts can be used as supervision to neural networks to solve the simpler tasks defined above and to recognize the structured event as well. In this case, neuro-symbolic frameworks e.g., DeepProbLog [13], Logic Tensor Networks [19], LYRICS [14] that combine low-level neural perceptions with logic reasoning (also know as symbolic) seem to be suitable approaches to achieve these objectives. In this paper, we propose a neuro-symbolic approach for structured event recognition from raw data that uses "shallow" annotation on the high-level events and exploits background knowledge to propagate this supervision to simpler tasks such as object classification. We develop a prototype of the approach and compare it with a purely neural solution based on a recurrent neural network, showing the higher capability of solving both the event recognition task and the simpler task of object classification, as well as the ability to generalize to events with unseen outcomes. The detailed contributions of the paper are the following:

1. a formal definition of the problem of structured event recognition from raw data sequences with "shallow" annotations and of a neuro-symbolic solution combining low-level neural-based predictions with high-level reasoning;
2. a framework for automatically generating simple videos containing events which are fully annotated;
3. a prototypical neuro-symbolic recognition approach based on DeepProbLog;
4. an experimental evaluation that compares our approach with a purely neural solution, showing the advantage of explicitly using background knowledge.

The rest of the paper is organized as follows: Sections 2 formally defines the problem of structured event recognition with "shallow" annotation; Section 3 presents our proposed solution; Section 4 briefly reviews the state of the art approaches that have been proposed in the context of event recognition, and presents some of the most well-known neuro-symbolic frameworks; Section 5 describes the event generation framework; Section 6 presents the experimental setting; Section 7 describes the neural LSTM approach and our prototype approach based on DeepProbLog; Section 8 reports the experimental results; Finally, Section 9 draws some concluding remarks and discusses directions for future work.

2 Problem definition

Let $\mathcal{L}$ be a first order language with three sorts, $\mathbb{O}$, $\mathbb{E}$, and $\mathbb{T}$. Terms of sort $\mathbb{O}$ denote objects, terms of sort $\mathbb{E}$ denote events, and terms of sort $\mathbb{T}$ denote time-points. The language contains the constants $0, 1, 2, \ldots$ of sort $\mathbb{T}$, used to name time points and the binary relation $<: \mathbb{T} \times \mathbb{T} \rightarrow \{\top, \bot\}$. The language $\mathcal{L}$ also contains a set of predicates $\mathcal{P}$ of sort $\mathbb{O}^k \rightarrow \{\bot, \top\}$, which are used to describe the time invariant properties and relations between objects. $\mathcal{L}$ contains also a set of function symbols $\mathcal{E}$ of sort $\mathbb{O}^k \rightarrow \mathbb{E}$ that are used to describe events that involve a (possible empty) tuple of objects. We also have a relationship $outcome(\mathbb{E}, \mathbb{O})$ that

is used to describe the fact that the outcome of an event is an object. Finally, $\mathcal{L}$ contains the predicate $happens(\mathbb{E}, \mathbb{T}, \mathbb{T})$ that is used to describe the fact that a certain event happens within an interval of time. For example, the formula $\exists x.happens(drop(John, x), t_1, t_2)$ states that *John* drops an object x at some time between t_1 and t_2. Notice that events can "create" new objects, for example the result of mixing milk and coffee is a cappuccino. This is expressed by the formula $milk(x) \wedge coffee(y) \rightarrow outcome(mix(x, y), z) \wedge cappuccino(z)$. A *narrative* is an interpretation $\mathcal{I}$ of the language $\mathcal{L}$, where the terms of sort $\mathbb{T}$ are interpreted in the set of natural numbers and $<$ in the usual linear order. The terms of sort $\mathbb{O}$ are interpreted in a domain of objects $\Delta_{\mathbb{O}}$ and those of $\mathbb{E}$ are interpreted in a domain of events $\Delta_{\mathbb{E}}$. Since we are interested in finite narratives, i.e., narratives that involve a finite number of objects and a finite number of events and time points, we can specify a narrative by using the Herbrand Base. In particular, for every $k > 0$ we define a k-narrative as a pair $\mathcal{N} = (\mathcal{C}, \mathcal{F})$ where:

- $\mathcal{C}$ is a finite set of new constants for objects of type $\mathbb{O}$;
- $\mathcal{F}$ is a subset of ground atoms in the language of $\mathcal{L}$ extended with the constants in $\mathcal{C}$ and the constants $0, 1, \ldots, k$ of sort $\mathbb{T}$; such that: if $happens(e, t_1, t_2) \in \mathcal{F}$ then $t_1 \leq t_2$.

Our main aim is to reconstruct a narrative from a data stream using some neuro-symbolic method that is capable of combining low-level data processing capabilities with the ability to leverage background knowledge about the structure of events. More formally, let $\boldsymbol{D} = \{\boldsymbol{d}_i\}_{i=1}^{k}$ be a data sequence of length k, where each $\boldsymbol{d}_i \in \mathcal{X}$ is a low-level representation for sequence element i (like a real-valued vector, matrix or tensor). Our main objective is to generate a k-narrative that describes the events that happen in $\boldsymbol{D}$, when they happens, their participants, and their outcomes. In other words we want to extract from $\boldsymbol{D}$:

- a set of objects;
- the properties and the relations between the objects;
- the set of events that happen;
- the objects (arguments) that are involved in each event that happens;
- the outcomes of the events that happen.

▶ **Example 1.** Let $\boldsymbol{D}$ be a video showing two people, one moving, leaving a bag and then moving away, and the other standing. We would like to produce the following narrative:

$$\mathcal{C} = \{p_1, p_2, b_1\} \qquad \mathcal{F} = \left\{ \begin{array}{c} person(p_1), person(p_2), bag(b_1), \\ happens(move(p_1), 0, 4), happens(drop(p_1, b_1), 4, 5) \\ happens(move(p_1), 5, 7), \end{array} \right\}$$

The type of supervision we suppose to have, in order to learn a model that extracts narratives from data, is partial and consists of a set of n data sequences labelled with *some* (not necessarily all the) ground facts about events happening in the sequence:

$$\left\{ \boldsymbol{D}^{(i)}, \mathcal{F}_p^{(i)} \right\}_{i=1}^{n}$$

where $\boldsymbol{D}^{(i)} = \{\boldsymbol{d}_j^{(i)}\}_{j=1}^{k}$ is a data sequence and $\mathcal{F}_p^{(i)}$ is a set of positive and negative literals, denoting a subset of the events that happen or don't happen in $\boldsymbol{D}^{(i)}$. Notice that we do not need to have a complete labelling for all the events. Furthermore, notice that the supervision provided via $\mathcal{F}_p^{(i)}$ also provide a supervision for the subset of objects $^{(i)}$ that appear in the data stream $\boldsymbol{D}^{(i)}$, which is the set of constants of type $\mathbb{O}$ that appear in the positive literals of $\mathcal{F}_p^{(i)}$.

Events can be related to each other, and structured events can be defined in terms of simpler ones.

▶ **Example 2.** Let *potential_threat* represent a structured event corresponding to a potential threat represented by something happening in a video like the one in the previous example. The threat could be modelled by the following formula:

$$\begin{aligned} &\mathit{happens}(\mathit{potential\_threat}, t_0, t_3) \leftrightarrow \\ &\qquad \exists x, y, t_1, t_2 . \mathit{person}(x) \wedge \mathit{bag}(y) \wedge \\ &\qquad\qquad \mathit{happens}(\mathit{move}(x), t_0, t_1) \wedge \\ &\qquad\qquad \mathit{happens}(\mathit{drop}(x, y), t_1, t_2) \wedge \\ &\qquad\qquad \mathit{happens}(\mathit{move}(x), t_2, t_3) \end{aligned} \tag{1}$$

An example of supervision in this context could be a set of videos $\boldsymbol{D}^{(1)}, \boldsymbol{D}^{(2)}, \ldots, \boldsymbol{D}^{(m)}$ of length k, each of which is annotated with either the single fact $\mathit{happens}(\mathit{potential\_threat}, 0, k)$ or with the single fact $\neg \mathit{happens}(\mathit{potential\_threat}, 0, k)$.

3 Proposed solution

Looking at the examples of the previous section, we observe that a structured event can be expressed in terms of simple events using logical languages. Simple events include the objects participating in the structured event, their relationships and their individual actions. Therefore, the correct recognition of the simple events combined with the definition of the structured event at the logical level will lead to the recognition of the structured event.
Our proposed approach has two aims, respectively learning to recognize the structured event happening in a data sequence and the simple events that compose it. To achieve these aims, we provide both background knowledge on the domain, expressed in terms of logical formulas, and "shallow" annotations on the structured event, like the one for the *potential_threat* example of section 2. In order to solve our problem, we have to complete three tasks:

object detection: in order to build the narrative, we have to find the set of objects $\mathcal{C}$[1] that appear in a data sequence $\boldsymbol{D}$.

object classification and relation detection: We also have to classify the objects in their types, e.g., a chair, a person, ..., and we have to detect relations between objects, e.g. if the person holds a bag or not.

event recognition: we have to recognize the events that happen in the video.

The traditional approach to solve the problem is to use a pipeline, where the above tasks are solved sequentially and the result of the solution of the previous task is provided as input to the next task. However, this requires supervision at all levels, the objects in the data, their class and the events. We instead have only partial supervision on some events.
In our solution we propose to have a fully end-to-end approach in which both the supervision on data and the background knowledge are used to train some neural networks for more data driven tasks such as object detection and classification. We therefore suppose to have the following components:

- A neural network $\mathrm{Det}_{\mathrm{nn}}$ that takes as input a sequence $\boldsymbol{D}$ and returns a set of objects $\mathcal{C}$ each of which is associated with a set of numeric features $f(o)$. For example, if $\boldsymbol{D} = \{\boldsymbol{d}_i\}_{i=1}^{k}$ is a video, then $f(o)$ contains the bounding boxes of object o at each frame $\boldsymbol{d}_i$ and the crop of the image on the bounding box for each frame;
- for some (not necessarily all) object predicates we have a network P_{nn} that takes as input the features of an n-tuple of objects $f(o_1), f(o_2), \ldots, f(o_n)$ and returns a sequence in $[0, 1]^k$ which represents the level of truth or probability of truth of the predicate at

[1] objects and constants are used interchangeably

each time point $0 \leq i \leq k$. For example, for the *person* predicate in the examples in the previous section we would have a network that given a sequence of cropped images outputs for each image the probability that it contains a person.

All the outputs of the neural networks defined above can be combined with the background knowledge which is described in terms of the axioms, such as equation (1). The way in which this combination is achieved could be based both on probabilistic semantics or on fuzzy semantics. At this stage we do not want to commit on one or the other. A list of neuro-symbolic approaches that can be adopted to implement our architecture is provided in the related work section. In the following we provide a proof-of-concept implementation of the architecture described above using DeepProbLog [13].

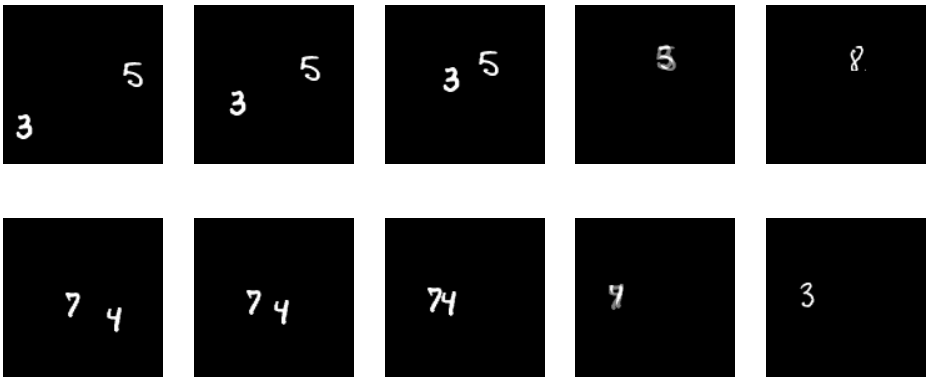
4 Related work

Event recognition from data streams like videos, audio and text is a well-studied problem. With the advent of deep learning, most sub-symbolic approaches to event recognition have moved from processing hand-crafted features to automated representation learning from raw data (see [1] and [22] for a survey). In order to be effective, however, these approaches require large and deeply annotated datasets, where supervision on the different sub-tasks defining the event recognition problem is available for training. On the other hand, purely symbolic approaches to event recognition [4] allow to explicitly define the conditions that lead to the occurrence of an event, but may fail in the presence of noise. As a consequence, symbolic approaches that can deal with uncertainty have emerged (see [2]). In [20, 3], the authors recognize structured events from a combination of low-level events. This recognition was done considering only uncertainty on low-level events and using ProbLog [17] as a probabilistic reasoner. Low-level predictions are however assumed to be given and no attempt is made at predicting low-level events from raw data. A number of approaches [9, 10, 23] combine symbolic reasoning with pre-trained neural networks used to recognize low-level events. These approaches require full supervision at the different levels and cannot be applied in the setting we address in this work.

In this work we aim at combining the advantages of low-level neural processing and high-level symbolic reasoning to achieve effective event recognition from small datasets and shallow annotations. Neuro-symbolic integration is an active area of research and multiple frameworks have been proposed. These frameworks combines low-level neural perception with high level reasoning in different ways. Many approaches, like LYRICS [14] or Logic Tensor Networks [19] combine neural predictors with fuzzy logic. NeurASP [24] combines neural networks with answer set programming, while [6] and [15], use neural networks to define potentials in probabilistic graphical models. Among existing frameworks, DeepProbLog [13] is particularly appealing in terms of expressivity. DeepProbLog extends the probabilistic programming language ProbLog with neural predicates, achieving an elegant and powerful combination of neural networks, logic and probability. We thus leverage DeepProbLog as the underlying integration framework for our neuro-symbolic event recognition prototype.

5 Event generation framework

Most of the available datasets for event recognition provide only limited annotations on some but not all the elements of an events. For example, in the context of event recognition in video there are dataset like Olympic Sport [16] and UCF101 [21] that provide annotation only on the event happening in the video. Datasets like CAVIAR [5], MEVA [7], Cooking [11] and



(a)

(b)

Figure 1 Example of events generated by the framework: join_add (a) and join_sub (b).

HiEve [12] provide a richer annotation, e.g., objects classes, object locations and distinction between simple and structured events, but they introduce a level of complexity in the visual part that requires pre-trained models for processing low level features. Here we are interested in developing a neuro-symbolic system that is trainable end-to-end, where the learning of the low-level processing is influenced by the high-level knowledge. Furthermore, the above mentioned datasets were manually curated, and cannot be extended to consider newly defined structured events without a tedious process of data collection and manual annotation. We, instead, would like to be able to quickly generate new data streams containing new events so as to support a fast prototyping and testing of recognition architectures. For these reasons, we have implemented a video generator of events involving mnist digits. The generator allows to generate videos of different length and with a different number of digits that interact with each other, together to the narrative describing the objects and events in the video. The generator uses an object predicate $digit(x, v)$ to indicate that v is the value corresponding to object x. Concerning events, we distinguish between simple events that involve single digits and structured ones that involve combinations of digits. The simple events we defined are:

- $appear(x)$: a digit x appears in the video
- $disappear(x)$: a digit x disappears from the video
- $enter(x)$: a digit x enters in the video
- $exit(x)$: a digit x exits from the video

The difference between $appear/disappear$ and $enter/exit$ is that in the former case the digit is always fully visible when in the video, while in the latter case the digit is only partially visible upon entering/exiting. Example of structured events definable in the framework are:

- $join\_add(x, y)$: two digits, respectively x and y, approach each other, overlap and then the digit that is the result of the sum of the two digits appears, i.e., $outcome(join\_add(x, y), z)$ with $digit(x, v_x), digit(y, v_y), digit(z, v_z)$ and $v_z = v_x + v_y$. Note that this event can only happen if the sum of the two digits is ≤ 9.
- $join\_sub(x, y)$: two digits, respectively x and y, approach each other, overlap and then the digit that is the result of the difference between the two digits appears, i.e., $outcome(join\_sub(x, y), z)$ with $digit(x, v_x), digit(y, v_y), digit(z, v_z)$ and $v_z = max(v_x, v_y) - min(v_x, v_y)$.
- $split(x)$: a digit x splits into two digits whose sum or difference gives the value of x, i.e., $outcome(split(x), (y, z))$ with $digit(x, v_x), digit(y, v_y), digit(z, v_z)$ and $v_x = v_y + v_z$ or $v_x = max(v_y, v_z) - min(v_y, v_z)$.

Some examples of structured events produced by the generator are shown in Figure 1. For simplicity, each object is assumed to participate in at most one simple event and one structured event for each frame. For each video, a narrative file is also produced that contains the following information for each digit:

- the name: a label
- the class: the corresponding mnist class
- the position: x and y coordinates inside the frame
- the simple event (if any) the digit is involved in
- the structured event (if any) the digit is involved in

6 Experimental setting

Our experimental evaluation is aimed at verifying whether a neuro-symbolic solution has an advantage in recognizing structured events with respect to a fully neural approach. In addition to the capability of correctly classifying each video into the corresponding structured event, we aim at evaluating the ability to learn to correctly classify the underlying objects (the digits) as well as the ability to generalize to unseen outcomes (e.g., the result of a `join_add` being a digit for which no explicit supervision was ever received). The scenario and learning setting we created to this aim are described in the following.

6.1 Scenario

The scenario consists of videos produced with the event generation framework described in Section 5. Each video consists of 10 frames, each frame showing one or two digits. Digits can appear anytime within the first half of the video, and only disappear if they join together. When present, the digits are always completely visible, apart from the frames in which they overlap with each other (e.g., right before a join). We generated three types of videos:

- *join_add*
- *join_sub*
- *no_join*

The first two refer to videos where the corresponding structured event as discussed in Section 5 takes place. The resulting digit can stay in the same position or move. To avoid ambiguities, we refrain from generating videos where one of the two operands is a zero. As consequence, the only way to get a zero is in a *join_sub* when both digits are equal. The third type refers to videos where neither of the two structured events takes place. In this case the video contains two arbitrary digits that wander around with no restrictions, possibly overlapping with each other.

6.2 Learning setting

Our goal is to test the ability of the different approaches to learn to recognize events with partial supervision. The idea is to provide supervision in terms of the structured event taking place (if any) and the outcome of the event (i.e., the result of the addition/subtraction). Supervision is thus provided in terms of sets like the following:

$$\{happens(join\_add(x,y),1,T), outcome(join\_add(x,y),z), digit(z,4)\}$$
$$\{happens(join\_sub(x,y),1,T), outcome(join\_sub(x,y),z), digit(z,2)\}$$
$$\{\neg happens(join\_add(x,y),1,T), \neg happens(join\_sub(x,y),1,T)\}$$

Note that this type of feedback provides information on the classification of the underlying objects, even if only when a join takes place, and only for the digit which is the result of the join. We thus build the task to additionally test the ability of the methods to *generalize* to unseen outcomes, i.e., digits that were never observed as the result of a join during training (or validation). For the sake of conciseness, in the following we will refer to the combination of structured event and outcome as the class of a video (with *no_join* being the class of a video where no join occurs). To generate the videos we first split the original MNIST dataset into training, validation and test set. Then, separately for each set, we randomly picked digits to generate a set of videos for each of the video classes, making sure that each class had the same number of videos. We generated training and validation videos containing *no_join*, *join_add* with outcome from 2 to 7 and *join_sub* with outcome from 0 to 7, for a total of 15 video classes. Test videos contain the same classes as the training and validation ones plus *join_add* with outcome 8 and 9 and *join_sub* with outcome 8, for a total of 18 classes. We generated 1500 videos for training, 150 for validation and 180 for testing, so that each class always contains 100 videos.

7 Event recognition approaches

In this section, we describe the learning approaches that we used to solve the aforementioned task. We start presenting the low-level neural networks that we use for object detection and classification and proceed describing the fully neural and the neuro-symbolic approaches that build on them. The object detector is pre-trained, while the object classifier is trained end-to-end both in the fully neural and neuro-symbolic approaches. Training is performed for 35 epochs, using the Adam optimizer with a learning rate of 0.001 and early stopping on the validation set. Training for more epochs does not lead to improvements in recognition quality.

7.1 Object detector and classifier

The object detector is a neural network that extracts (processed) patches from frames. Its architecture is shown in Figure 2 for the case of a single frame, as the same structure is repeated for all frames in a video. Its main module is a standard convolutional neural network that consists of two convolutional layers, each followed by a max-pooling layer, and two fully-connected layers. ReLU are used as activation in all layers apart from the output layer where a sigmoid is used. The module takes as input a frame of size 128×128 and gives as output a vector of length 6:

$$\boldsymbol{o}_{det} = \langle v_1, v_2, x_1, x_2, y_1, y_2 \rangle$$

where $v_i \in [0, 1]$ indicates whether the i-th digit is present in the frame and $x_i, y_i \in [0, 1] \times [0, 1]$ are the normalized digit coordinates (digits are ordered according to the distance between their predicted coordinates and the origin). The two patches corresponding to the coordinates are extracted from the frame, and their content is multiplied by the value of their visibility flag. In so doing, the detector outputs "soft" patches, that depending on the value of the visibility flag range from the patch itself ($v_i = 1$) to a completely black patch ($v_i = 0$).

The digit classifier has the same architecture as the main module of the detector, with the sigmoid replaced by a softmax in the output layer. The classifier takes as input an image of size of 28×28 which corresponds to a processed patch extracted by the detector and returns as output a vector of length 11, where the first 10 element refers to the 0-9 digits and the last one indicates the absence of a digit. This module is repeated for all patches and all frames of the input video.

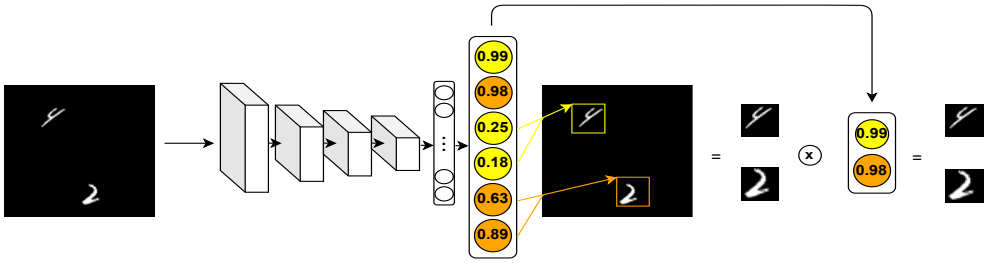


Figure 2 Mnist digit detector.

7.2 Fully neural approach

The fully neural approach combines the predictions of the digit detector and classifier on the different frames using an LSTM recurrent neural network [8]. The overall architecture is shown in Figure 3. For each frame in the input video, the detector extracts a pair of patches and sends them to the digit classifier. The predictions of the classifier are concatenated with the visibility and coordinate predictions from the detector and fed to an LSTM cell. After processing the entire input sequence, the LSTM outputs a prediction in three classes, $join\_add, join\_sub$ and no_join. The outcome of the join event is recovered from the output of the digit classifier on the first patch of the last frame. If the class with the highest prediction is no_join, the outcome prediction is ignored.

7.3 Neuro-symbolic approach

We developed a neuro-symbolic approach for structured event recognition using the DeepProbLog [13] framework. This framework can be seen as a neural extension of the probabilistic extension of Prolog, ProbLog [17]. Like ProbLog, the knowledge about the domain is encoded as a set of logical rules (i.e., horn clauses). In addition, DeepProbLog introduces neural predicates that allow to instantiate facts as outputs of neural predicates processing raw data. The neural extension is realized by enhancing ProbLog with a primitive that allows to declare neural predicates:

$$nn(n_{id}, \boldsymbol{X}_s, Y, \boldsymbol{y}_s)$$

where nn is a reserved functor used to declare a neural predicate, n_{id} is an identifier for the underlying neural network, $\boldsymbol{X}_s$ denotes a sequence of n input variables, Y is the output variable, and $\boldsymbol{y}_s$ denotes a sequence of m possible values that Y can assume. Training of these neural predicates is done by providing supervision on the head of the logical rules expressed as standard logical queries. This means that in our prototype the "shallow" annotations on the structured event will be mapped to queries, while simple events will be mapped to neural predicates.

The DeepProbLog program we defined to address the event recognition task is shown in Figure 4. It consists of the following predicates:

- $digit(X, V, T, V_x)$: a neural predicate that states that the X digit of video V at time T is V_x
- $join\_add\_res(V, V_z)$: a binary predicate that states that video V is a $join\_add$ and the resulting digit of the join is V_z
- $join\_sub\_res(V, V_z)$: a binary predicate that states that video V is a $join\_sub$ and the resulting digit of the join is V_z
- $no\_join(V)$: a unary predicate that states that video V is a no_join video

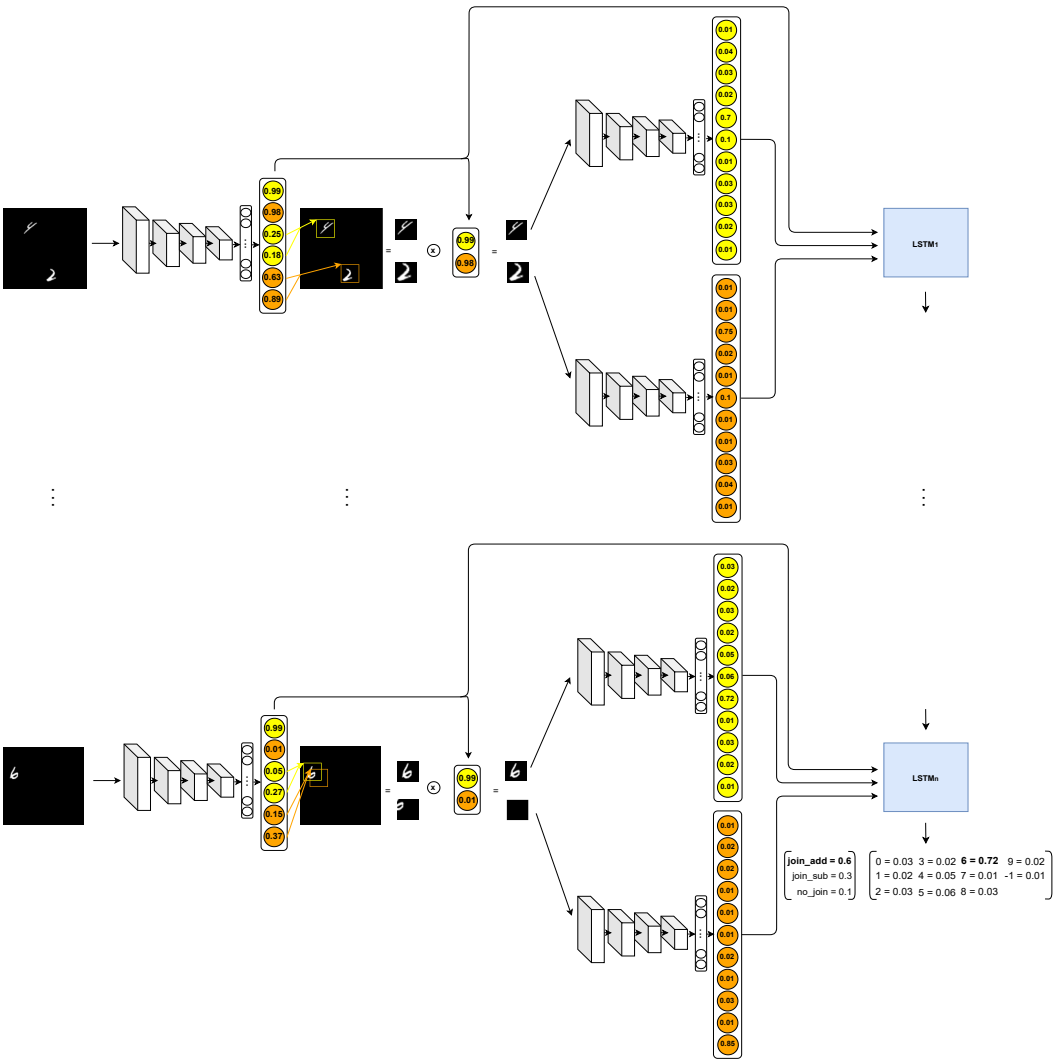


Figure 3 Fully neural approach: LSTM-based architecture.

The neural predicate $digit(X, V, T, V_x)$ is mapped to the combination of digit detector and classifier shown in Figure 3, with the difference that only the output of the classifier (i.e., a probability distribution on the 0-9 digits plus the absence of the digit) is provided. The predicate $join\_add\_res(V, V_z)$ basically represents the combination of $join\_add(X, Y)$, $outcome(join\_add(X, Y), Z)$ and $digit(Z, V_z)$, with the addition of the V variable indicating the video (omitted for simplicity in the formalization throughout the paper). The predicate checks whether there are two digits in the first half of the video and only one digit at the end that is the sum of the two. The $join\_sub\_res$ predicate is similar to $join\_add\_res$ with sum replaced by difference (in absolute value, so that digits do not need to be sorted). Finally, for a no_join, we know that there are two digits for the whole duration of the video. Therefore, we define a rule that only fires when both digits are visible in the last frame.

```
nn(mnist_net, [I, V, T], Y, [0,1,2,3,4,5,6,7,8,9,-1]) :: digit(I, V, T, Y).

join_add_res(V, Z) :-
    between(0, 4, T1),
    digit(0, V, T1, X),
    X > 0, X < 9,
    digit(1, V, T1, Y),
    Y > 0, Y < 10 - X,
    digit(0, V, 9, Z),
    Z is X + Y, Z > 1,
    digit(1, V, 9, -1).

join_sub_res(V, Z) :-
    between(0, 4, T1),
    digit(0, V, T1, X),
    X > 0,
    digit(1, V, T1, Y),
    Y > 0,
    digit(0, V, 9, Z),
    Z is abs(X-Y),
    digit(1, V, 9, -1).

no_join(V) :- digit(1, V, 9, X), X =\= -1.
```

Figure 4 Neuro-symbolic approach: DeepProbLog program.

8 Results

In this section, we present and compare the results of the neural based LSTM approach with our proposed neuro-symbolic approach based on DeepProbLog on the tasks defined in Section 6.

Confusion matrices, where entries (i, j) denote the number of samples of true class i classified as class j, for the event recognition problem and related sub-problems for the two approaches are shown in Figure 5. The first row shows the confusion matrices for the event and outcome recognition (*join_add* with outcome from 1 to 9, *join_sub* with outcome from 0 to 8, *no_join*), while the second row reports the confusion matrices for the underlying task of digit classification (0-9, and -1 corresponding to no digit). The left column reports results for the fully neural approach, the right column those for the neuro-symbolic approach.

Looking at the top left confusion matrix, we can observe that the neural approach is able to recognize the events for which the supervision is provided, even if it sometimes mistakes a *join_add* for a *join_sub* and vice-versa when the outcome is the same. On the other hand, it completely fails in generalizing to unseen events (*join_add* with outcome 8 or 9, *join_sub* with outcome 8). This fact highlights the difficulty of the neural approach in fully learning the semantic behind the join operations. The results of the confusion matrix on digits (bottom left) confirm these findings, as the network fails to classify digits for which no direct supervision is available (i.e., 8 and 9).

The situation with our neuro-symbolic approach is rather different (right column). Indeed, DeepProbLog is capable of predicting the unseen outcomes with reasonable accuracy, and the same holds for the underlying digit classification task. If we compare the confusion matrices on the digits of the two approaches (bottom row), we can observe that our approach has a higher accuracy even on digits for which direct supervision is available. These results clearly indicate the importance of the background knowledge in compensating partial supervision and allowing to generalize beyond what is observed during training.

9 Conclusion and future work

In this work, we have proposed a neuro-symbolic approach for structured event recognition from data sequences, where background knowledge about event structure is combined with deep neural networks used to solve the sub-tasks of event recognition such as object detection

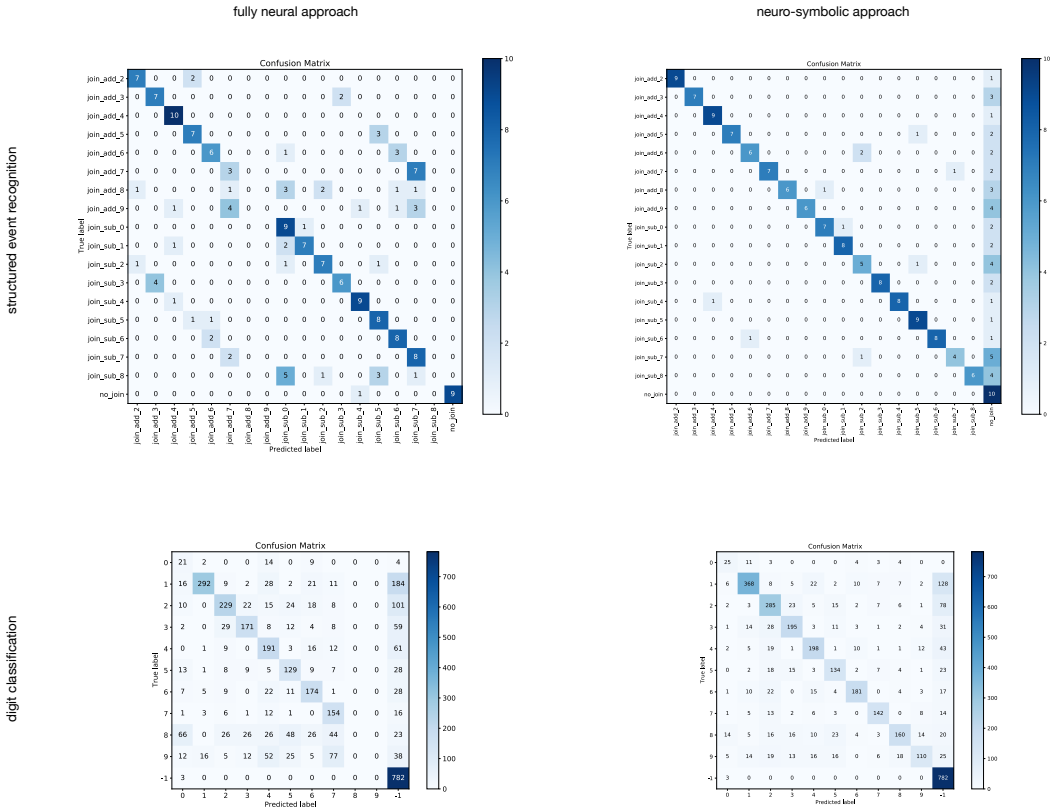


Figure 5 Experimental results: confusion matrices for event + outcome recognition (top row) and digit classification (bottom row). Left: fully neural approach; right: neuro-symbolic approach.

and classification. The proposed architecture can be trained end-to-end with data streams containing only shallow annotations. We prototyped our architecture using DeepProblog as a neuro-symbolic integration framework and tested it on a structured event recognition problem defined on a synthetic dataset automatically generated. The experiments show that the background knowledge about structured events and their outcomes translates supervision on the structured event into supervision on lower-level predictive tasks like object classification, allowing to successfully train the neural components of the architecture. We compare our architecture with a purely neural solution that uses the same basic components for object detection and classification. The comparison shows how the use of background knowledge improves performance for both high-level and low-level prediction tasks. The advantages of these effects are multiple. The first advantage is the fact that we are able to train a classifier without a direct supervision on some of the classes (the classes 8 and 9 in our experiment); a second advantage concerns explanability: while in a fully neural approach it is not possible to explain the happening of an event in terms of its components (object participants, and their types), in our approach the reasoning process that infers the happening of a structured event on the basis of the recognition of some basic facts (detection of an object of a certain type) can be provided as an explanation. As future work, we plan to test the implementation of the proposed architecture on different neuro-symbolic frameworks, and to consider more structured events and also the application of the methodology on real data.

References

1 Kashif Ahmad Ahmad and Nicola Conci. How Deep Features Have Improved Event Recognition in Multimedia: A Survey. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 15(2):39:1–39:27, 2019. doi:10.1145/3306240.

2 Elias Alevizos, Anastasios Skarlatidis, Alexander Artikis, and Georgios Paliouras. Probabilistic Complex Event Recognition: A Survey. *ACM Computing Surveys*, 50(5):71:1–71:31, 2017. doi:10.1145/3117809.

3 Alexander Artikis, Evangelos Makris, and Georgios Paliouras. A probabilistic interval-based event calculus for activity recognition. *Annals of Mathematics and Artificial Intelligence*, 89(1-2):29–52, 2021. doi:10.1007/s10472-019-09664-4.

4 Alexander Artikis, Anastasios Skarlatidis, François Portet, and Georgios Paliouras. Logic-based event recognition. *The Knowledge Engineering Review*, 27(4):469–506, 2012. doi:10.1017/S0269888912000264.

5 Caviar: context aware vision using image-based active recognition, 2011. URL: http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1.

6 Liang-Chieh Chen, Alexander Schwing, Alan Yuille, and Raquel Urtasun. Learning Deep Structured Models. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1785–1794. JMLR.org, 2015. URL: http://proceedings.mlr.press/v37/chenb15.html.

7 Kellie Corona, Katie Osterdahl, Roderic Collins, and Anthony Hoogs. MEVA: A Large-Scale Multiview, Multimodal Video Dataset for Activity Detection. In *IEEE Winter Conference on Applications of Computer Vision, WACV 2021, Waikoloa, HI, USA, January 3-8, 2021*, pages 1059–1067. IEEE, 2021. doi:10.1109/WACV48630.2021.00110.

8 Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997. doi:10.1162/neco.1997.9.8.1735.

9 Abdullah Khan, Loris Bozzato, Luciano Serafini, and Beatrice Lazzerini. Visual Reasoning on Complex Events in Soccer Videos Using Answer Set Programming. In Diego Calvanese and Luca Iocchi, editors, *GCAI 2019. Proceedings of the 5th Global Conference on Artificial Intelligence, Bozen/Bolzano, Italy, 17-19 September 2019*, volume 65, pages 42–53. EasyChair, 2019. doi:10.29007/pjd4.

10 Abdullah Khan, Luciano Serafini, Loris Bozzato, and Beatrice Lazzerini. Event Detection from Video Using Answer Set Programing. In Alberto Casagrande and Eugenio G. Omodeo, editors, *Proceedings of the 34th Italian Conference on Computational Logic, Trieste, Italy, June 19-21, 2019*, volume 2396 of *CEUR Workshop Proceedings*, pages 48–58. CEUR-WS.org, 2019. URL: http://ceur-ws.org/Vol-2396/paper25.pdf.

11 Paula Lago, Shingo Takeda, Sayeda S Alia, Kohei Adachi, Brahim Bennai, and Sozo Inoue Francois Charpillet. A dataset for complex activity recognition withmicro and macro activities in a cooking scenario. *CoRR*, abs/2006.10681, 2020. arXiv:2006.10681.

12 Weiyao Lin, Huabin Liu, Shizhan Liu, Yuxi Li, Rui Qian, Tao Wang, Ning Xu, Hongkai Xiong, Guo-Jun Qi, and Nicu Sebe. Human in Events: A Large-Scale Benchmark for Human-centric Video Analysis in Complex Events, 2020. arXiv:2005.04490.

13 Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc D. Raedt. DeepProbLog: Neural Probabilistic Logic Programming. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, volume 31, pages 3753–3763, 2018. URL: https://proceedings.neurips.cc/paper/2018/hash/dc5d637ed5e62c36ecb73b654b05ba2a-Abstract.html.

14 Giuseppe Marra, Francesco Giannini, Michelangelo Diligenti, and Marco Gori. LYRICS: A General Interface Layer to Integrate Logic Inference and deep Learning. In Ulf Brefeld, Élisa Fromont, Andreas Hotho, Arno J. Knobbe, Marloes H. Maathuis, and Céline Robardet, editors, *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2019, Würzburg, Germany, September 16-20, 2019, Proceedings, Part II*, volume 11907 of *Lecture Notes in Computer Science*, pages 283–298, 2019. doi:10.1007/978-3-030-46147-8_17.

15 Giuseppe Marra, Francesco Giannini, Michelangelo Diligenti, and Marco Gori. Integrating Learning and Reasoning with Deep Logic Models. In Ulf Brefeld, Élisa Fromont, Andreas Hotho, Arno J. Knobbe, Marloes H. Maathuis, and Céline Robardet, editors, *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2019, Würzburg, Germany, September 16-20, 2019, Proceedings, Part II*, volume 11907 of *Lecture Notes in Computer Science*, pages 517–532. Springer, 2020. doi:10.1007/978-3-030-46147-8_31.

16 Juan C. Niebles, Chih-Wei Chen, and Li Fei-Fei. Modeling Temporal Structure of Decomposable Motion Segments for Activity Classification. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision - ECCV 2010, 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part II*, volume 6312 of *Lecture Notes in Computer Science*, pages 392–405. Springer, 2010. doi:10.1007/978-3-642-15552-9_29.

17 Luc D. Raedt, Angelika Kimmig, and Hannu Toivonen. ProbLog: A Probabilistic Prolog and Its Application in Link Discovery. In Manuela M. Veloso, editor, *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pages 2462–2467, 2007. URL: http://ijcai.org/Proceedings/07/Papers/396.pdf.

18 Fabrício Henrique Rodrigues and Mara Abel. What to consider about events: A survey on the ontology of occurrents. *Applied Ontology*, 14(4):343–378, 2019. doi:10.3233/AO-190217.

19 Luciano Serafini and Artur d'Avila Garcez. Learning and Reasoning with Logic Tensor Networks. In Giovanni Adorni, Stefano Cagnoni, Marco Gori, and Marco Maratea, editors, *AI*IA 2016: Advances in Artificial Intelligence - XVth International Conference of the Italian Association for Artificial Intelligence, Genova, Italy, November 29 - December 1, 2016, Proceedings*, volume 10037 of *Lecture Notes in Computer Science*, pages 334–348, 2016. doi:10.1007/978-3-319-49130-1_25.

20 Anastasios Skarlatidis, Alexander Artikis, Jason Filippou, and Georgios Paliouras. A probabilistic logic programming event calculus. *Theory and Practice of Logic Programming*, 15(2):213–245, 2015. doi:10.1017/S1471068413000690.

21 Khurram Soomro, Amir R. Zamir, and Mubarak Shah. UCF101: A dataset of 101 Human Actions Classes From Videos in The Wild. *CoRR*, abs/1212.0402, 2012. arXiv:1212.0402.

22 Wei Xiang and Band Wan. A Survey of Event Extraction From Text. *IEEE Access*, 7:173111–173137, 2019. doi:10.1109/ACCESS.2019.2956831.

23 Tianwei Xing, Marc R. Vilamala, Luis Garcia, Federico Cerutti, Lance Kaplan, Alun Preece, and Mani Srivastava. DeepCEP: Deep Complex Event Processing Using Distributed Multimodal Information. In *IEEE International Conference on Smart Computing, SMARTCOMP 2019, Washington, DC, USA, June 12-15, 2019*, pages 87–92. IEEE, 2019. doi:10.1109/SMARTCOMP.2019.00034.

24 Zhun Yang, Adam Ishay, and Joohyung Lee. NeurASP: Embracing Neural Networks into Answer Set Programming. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 1755–1762. ijcai.org, 2020. doi:10.24963/ijcai.2020/243.