A Multi-Agent-System framework for flooding events

Andrea Rafanelli[1,2,*], Stefania Costantini[2] and Giovanni De Gasperis[2]

[1]Department of Computer Science, University of Pisa, Italy
[2]Department of Information Engineering, Computer Science and Mathematics, University of L'Aquila, Italy

Abstract

This paper presents the potential capabilities offered by an integrated multi-agent system comprising logical agents and a neural network, specialized in monitoring flood events for civil protection purposes. Here we describe the idea of a framework – at the moment only partially developed – consisting of a set of intelligent agents, which perform various tasks and communicate with each other to efficiently generate alerts during flood crisis events, collaborating with a neural network, derived from the PSP-Net model, which is dedicated to the inspection and analysis of satellite images.

Keywords

Multi-Agent-System, DALI, Logic Programming, PSP-Net, Symbolic Sub-Symbolic integration

1. Introduction

The amount of damage caused by weather has increased dramatically in recent years. Current worldwide weather conditions are dramatically intensifying, resulting in an increase in storms and heavy rain precipitations[1]. This component is a significant contributor to global flooding issues. One of the most important feature of these severe events is the efficiency and efficacy with which these disasters can be identified and addressed. Immediate broadcast of alerts to population and delivery of emergency aid are essential. Deep Learning (DL) approaches are often used for image monitoring and analysis since they enable the extraction and classification of visual characteristics with incredible precision. However, in order to accomplish this, such systems require a substantial quantity of high-quality data. Unfortunately, the amount of data available concerning natural disasters is limited. In fact, these data are generated from past phenomena, which makes them hardly generalizable, especially considering that images taken from previous events have characteristics that are closely connected with the local region where the incident took place. The complexity of flood phenomena necessitates the development of reliable meteorological monitoring applications. Obtaining precise models capable of identifying these situations is therefore a fervent endeavour. Such models must be incorporated into systems that permit not only the detection of the problem but also, if possible, its reporting to the appropriate authorities in order to ensure a rapid and effective

WOA 2022: 23rd Workshop From Objects to Agents, September 1–2, Genova, Italy
*Corresponding author.
andrea.rafanelli@phd.unipi.it (A. Rafanelli); stefania.costantini@univaq.it (S. Costantini); giovanni.degasperis@univaq.it (G. De Gasperis)
0000-0001-8626-2121 (A. Rafanelli); 0000-0002-5686-6124 (S. Costantini); 0000-0001-9521-4711 (G. De Gasperis)

[1]https://www.epa.gov/climate-indicators/weather-climate

protection of the civilian population. In this context, hybrid systems permit the exploitation of a variety of capabilities derived from fundamentally distinct methodologies. Combining several techniques results in the construction of potentially highly specialised systems that can enhance the precision and accuracy of one-of-a-kind models [1]. Our long-term objective is to develop an agent-based system that combines the capabilities of neural networks with those of intelligent logical agents to obtain a tool for real-time flood recognition , alert transmission and mobilisation of rescue activities, fire departments, and essentially all required authorities. This study has the following specific objectives:

1. Use domain-specific agents of interest and create the conceptual architecture of the system;
2. Integrate image analysis with the decision-making capabilities of intelligent agents that can exhibit both reactivity and proactivity in emergency contexts;
3. Implement, in the future, the system, testing and validation through simulations;

Consequently, this paper is organised as follows: In Section 2, we discuss the approaches used to construct the multi-agent system and the neural network; in Section 3, we define the structure of our system and all of its components; in Section 4, we present the results of the experiment performed with the PSP-Net; and finally, in Section 5, we report the conclusions of the work.

2. Methods

This Section defines and outlines the main techniques used to develop our system. These methods are employed to generate the system's agents and to construct the neural network. A brief description of these approaches is then given below.

2.1. DALI

The material reported in this section is drawn from [2][3]. DALI is a logic programming language and development framework that allows to describes logical multi-agent system. It is a Prolog extension and has a fully logical semantics [4]. Following the description provided by [2], the formulation of a DALI logic program is described below. A DALI agent is a logic program that incorporates reactive rules, which are designed to interact with the external environment. External, internal, present, and previous events activate the reactive and proactive behaviour of a DALI agent. Syntactically, the prefix $\mathcal{E}$ represents external events. A reactive rule defines the response to each external event, using the token $:>$. The agent recalls reacting by transforming an external event into a previous event (prefix $\mathcal{P}$). An event perceived but not yet reacted to is designated by the prefix $\mathcal{N}$ as a "present event". Internal events render the DALI agent proactive and independent of its environment, enabling it to adapt and update its knowledge. Internal events consist of two rules and are represented by the postfix $\mathcal{I}$. The first rule comprises the necessary conditions for the reaction (second rule) to occur. Internal events are automatically attempted at a frequency that can be modified through directives in the startup file. User directives can alter numerous factors, including the frequency with which an agent should attempt internal events and the frequency with which it should respond to an

internal event. Finally, the postfix $\mathcal{A}$ represents actions. These can have preconditions or not; if they do, they are defined by action rules, whereas if they do not, they are simply action atoms. Similar to events, actions are recorded as past actions.
Formally, a DALI agent is defined by a tuple: $\mathcal{D}_{ag} = \{\mathcal{PD}_{ag}, \mathcal{E}, \mathcal{I}, \mathcal{A}\}$ where $\mathcal{PD}_{ag}$ is the agent's program, $\mathcal{E} = (E_1, ..E_n)$ is the set of external events, $\mathcal{I} = (I_1, ...I_m)$ is the set of internal events, and $\mathcal{A} = (A_1, ...A_s)$ is the set of actions.
In our MAS model, agents communicate according to the *Agent Communication Language* (ACL) compiled in the FIPA-ACL protocol[2]. A three-layers communication architecture has been incorporated into the DALI language. The first layer consists of a FIPA-compliant communication protocol and a communication filter, which is a set of criteria that determines whether a message can be sent and/or recevied (tell/told rules). The DALI communication filter is provided by rules that establish unique tell and told predicates at the meta-level. The second level adds a meta-level of reasoning that aims, if possible, to comprehend the content of communications based on ontologies. The third level is represented by the DALI interpreter. In addition, ASP-DALI[3] and Koinè DALI [5] are two significant DALI interface extensions that we want to adopt. ASP-DALI is a plugin for invoking ASP (Answer Set Programmin) solvers and executing ASP modules during the ASP DALI event. KOINE DALI is an extended framework that is compatible with Docker, Redis key/value data store and message broker, and other messaging technologies.

2.2. PSP-Net

Image semantic segmentation models organises pixels in a semantically relevant fashion, so pixels belonging from the same object class are classified together, assigning a common color or gray level. Typically, semantic segmentation models consist of two components: an encoder and a decoder. The encoder is responsible for extracting image features, while the decoder uses the extracted features to generate the segmented image.

The PSP-Net encoder is equipped with a CNN and a pyramidal pooling module (see Figure 1). The pyramidal pooling module is the primary component of this model, as it aids in capturing the image's global context by combining characteristics at several scales. This technique involves pooling the feature map, generated by the previous CNN, into different dimensions and then up-sampling the pooled features to make them the same size as the original feature map. The maps are then concatenated with the original feature map and sent to the decoder. The decoder then turns these features into predictions by passing them through its layers. The decoder is merely an additional network that transforms the characteristics into predictions.

We adopt the PSP-Net model to implement the semantic image segmentation module .

2.3. Translation module

The translation module is responsible for translating the neural network's output into an ASP program consisting of ground facts. The approach mentioned to is the one proposed in [7], in which a *detect module* is used to convert the image contents into an ASP program. The module is also used to segment the image and compute the positions of objects within it using

[2]cf. http://www.fipa.org/specs/fipa00037/SC00037J.html for language specification, syntax and semantics.
[3]https://github.com/AAAI-DISIM-UnivAQ/ASP_DALI

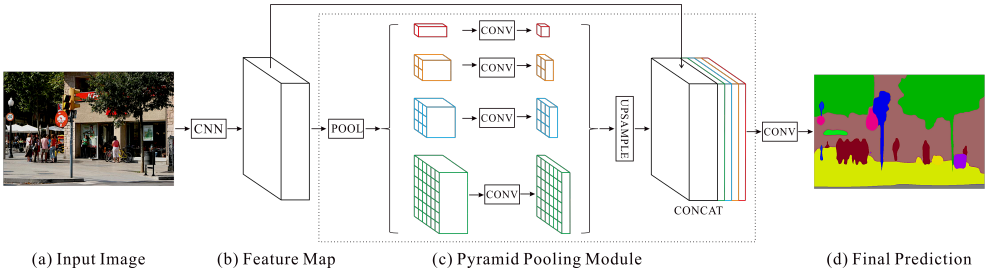

Figure 1: PSP-Net overview, source [6]

centroids and "basic arithmetic-based heuristics over bounding boxes to determine (spatial) entity relationships in the scene". Similarly, our objective is to convert the contents of an image into an ASP program finding the radius (the length of segments), and the positions of objects with 2D coordinates corresponding to the centroids of objects in aerial/satellite images. In this way, analyzing the respective angles between segmented masks centroids, we can derive their relative position and spatial relation. Centroids and radius will be used to get positions of grounded facts regarding object classes, i.e. : **road/3., water/3., building/3.**, etc.; but also to discover their relative spatial relationships such as:

- **inside(X,Y,R).,**
- **surrounded(X,Y,R).,**
- **beside(X,Y,R).,**
- **outside(X,Y,R).,**
- **adjacent(X,Y,R).,**
- **on_top(X,Y,R), on_left.., on_right, etc..**

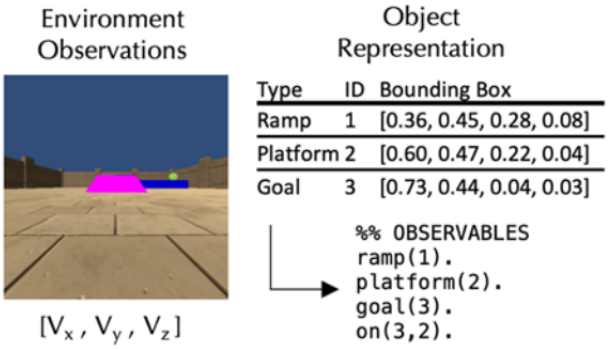

Figure 2: Detect module, source [7]

This grounded facts will constitute the logical factual description of a given satellite image, describing spatial relations between its object classes constituents.

3. FloodMAS framework

The system we present in this proposal is illustrated in Figure 3 and involves the interaction and synergy of several components. **FloodMAS** is a framework that integrates the capabilities of Neural Networks and a Multi-Agent System (MAS). For the implementation of our architecture, we will employ *Koinè DALI* and ASP-DALI. Koinè enables data and communication exchanges via the *Redis*[4] messsgr broker. Redis technology is used for data storage, delivery, and event handling. The MAS gets data via Redis subscribing to input topic channels and provides its response back to Redis publishing to other perception topic channels.

Specifically, our system will have an administrator named *Communication Agent* who will be responsible for managing event communications from external sources such as satellite photos and meteorological data, as well as neural network outputs.

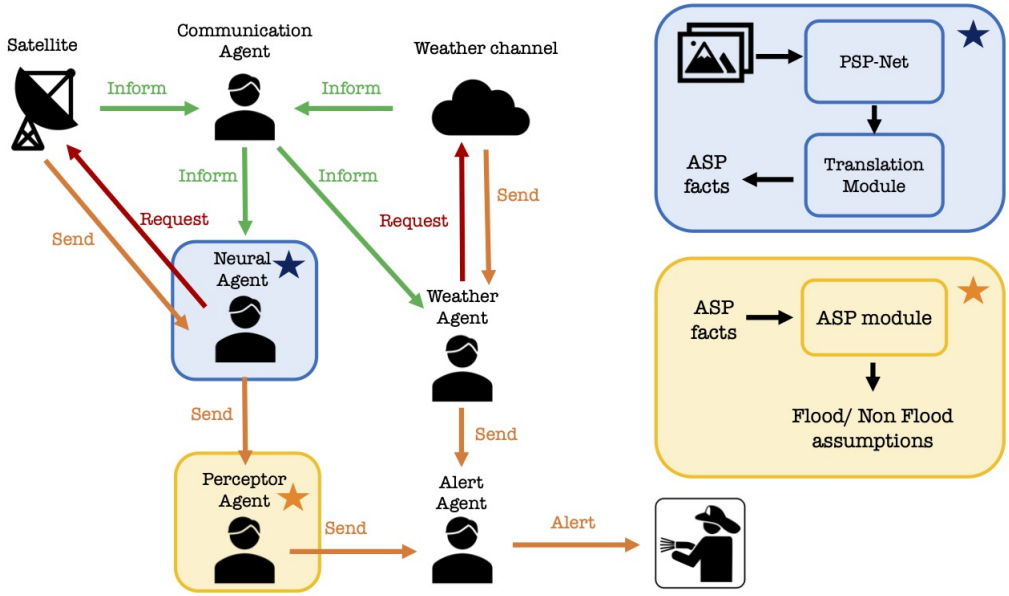

Figure 3: FloodMAS overview

3.1. Agents description

In the following,s we provide an overview of the methodology designed to implement our agent system and then provide a brief description of each agent, describing the thought roles, events, and possible actions.

(i) Communication agent

Communication agent is the agent responsible for initial agent synchronisation while bootstrapping and managing event communications from external sources. Initial agent synchronisation entails *Communication agent* collecting the external event *isReady* from all agents participating

[4]for more information on Redis, visit https://redis.io/

in the MAS, and then sending a departure message when each agent has indicated its presence.

(ii) Neural agent

Neural agent is the agent responsible for receiving, processing, and extracting logical predicates from satellite images. It obtains satellite photos through the provider's RESTful API. The API is translated into Koinè-compatible Redis events. This allows us to include data from external sources into our MAS. Neural agent utilises the PSP-Net network to retrieve the segmented mask of the image. The translation module is responsible for translating the output of the neural network into an ASP program consisting of ground facts (see Section 2.3). Thus, we will acquire a set of facts that will be conveyed to the *Perceptor agent*, which is responsible for reasoning about the facts (or logical predicates) retrieved from the neural network .

(iii) Perceptor Agent

Perceptor agent is the agent delegated to interpret satellite images, optical and infrared, producing their computational logic descriptions. It is connected to an ASP extension module that attempts to determine the image's content. Essentially, it uses the information it receives from *Neural agent* to identify whether or not there are flooding issues. This agent's ASP program comprises a set of rules to provide common-sense reasoning to the system. It has to deliver logical inferences derived from the answer sets solver. Since the ASP solver, as a modal logic inference engine, produces many possible answer sets derivable from the input facts, a selection algorithm based a specialized heuristic metric is then applied.

In a way, we want to expand the system's knowledge so that it can convey whether or not there is an actual flood event. Consider the following ASP program example:

flooded(building(X, Y, R)) :- building(X, Y, R), sourrounded(X, Y, R).
sourrounded(X, Y, R) :- water(V, W, R1), X+R $\leq$ V+R1, Y+R $\leq$ W+R1.

To determine whether a building is flooded from the exterior, it will likely be surrounded by water.

The common-sense rules that we intend to include at this point will be those that, given the extracted ground facts, enable the reasoner to form hypotheses about the image's content, and in particular to determine whether or not flooding is occurring in a certain location, excluding impossible spatial relations, and exploting a priori knowledge about how water physically tends to occupy the free spaces of homes and buildings were people may found themselves entrapped. Once the information on flooding or non-flooding has been retrieved from the picture, it is transferred over Redis to the *Communication agent*, which is responsible for forwarding the information to the *Alert agent.*

(iv) Weather agent

It is the agent's responsibility to manage weather information coming in from external authoritative and certified sources. This agent takes data from an external weather source and analyses it to determine whether there are any severe weather conditions that needs to be reported. Assuming that the agent receives weather data similarly to that shown in Figure 4, an example of information processing could be:

red_aler(X, Y) :- heavy_rain(X, Y).
heavy_rain(X, Y) :- assert(danger(X, Y, 1)), assert(meteo_danger).
meteo_dangerI :> send_message(alertAgent(X,Y)).

In this example, when the agent receives "red_alert" information, it generates an internal event that permits an alarm signal to be delivered to the *Alert agent.*

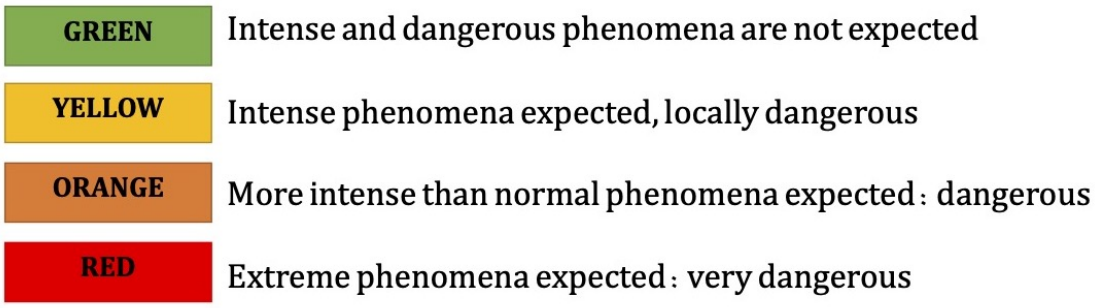

Figure 4: Weather information example

(v) Alert agent
This agent focuses in combining the messages received from the *Perceptor agent* and from the *Weather agent*. It is responsible for determining whether the messages are consistent and, if so, communicating with the appropriate authorities. In this situation, there are different potential outcomes:

- **Alert**: *Perceptor agent* and *Weather agent* both inform that there are flooding problems in the area of interest and, in this case, *Alert agent* notifies the authorities in real time through the **"send_message"** action;
- **Pre-alert**: either agent provides notification of a possible flooding problem and, in this case, *Alert agent* notifies the authorities through the **"inform"** action.

The difference between the two actions is that, in our opinion, if both agents agree on the presence of flooding events, the situation has a higher probability of occurring, and the Alert agent must therefore alert the relevant authorities; if, on the other hand, only one of the two agents reports the presence of a disaster, the probability of flooding will be significantly lower, and the agent notifies the authorities, who will ultimately decide whether or not the phenomenon is occurring in the indicated area.

4. PSP-Net experiment

In this Section, the experiment and its findings are described. We considered employing an Aerial Imagery data-set to forward our research. FloodNet [8] is the reference data-set. This set of data contains a small amount of pictures (2343 to be exact). The annotated classes provided are ten, as follows: 1) Background, 2) Flooded Building, 3) Non- Flooded Building, 4) Flooded Road, 5) Non-Flooded Road 6) Water, 7) Tree, 8) Vehicle, 9) Pool, and 10) Grass.

For our experiment, we decided to change the class distribution by removing the differentiation between flood and non-flood, as we want to leave the network with only the ability to recognize objects within the image and not the ability to determine whether there is a flooding event or not. Moreover, given the limited number of data, it was decided to increase the amount of data by rotating each image by 90, 180, 270 degrees. The dataset was then divided into train

(70%) and test (30%) by balancing for the different classes. A batch size of 8 was used and the model[5] was trained for 300 epochs with a learning rate of 1e-1 and a momentum of 1e-9. An SGD optimizer was chosen. Intersection-Over-Union (IoU), a standard evaluation metric for semantic image segmentation, was employed as the performance metric. As can be seen in the Figure 5, training phase accuracy reached 77%, whereas testing phase accuracy was roughly 66%. The model displayed modest overfitting (see also the figures representing the loss values in the train and test).

Two examples of network outputs are shown in Figure 6. It can be seen that the model produces quite reliable results even though the contours of the objects in the produced mask are not very well defined. In the future, we would like to test additional models, such as using the library *Detectron2*[6], as well as change some of the neural network's hyperparameters (loss, batch size, training time, initialization of weights, etc.)

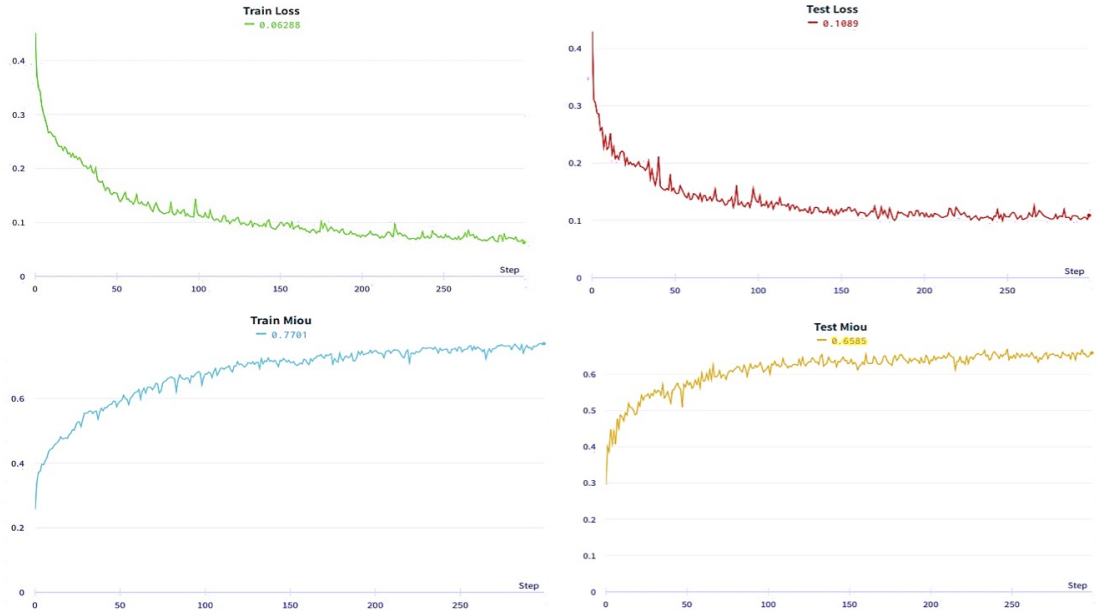

Figure 5: Performance metrics

5. Conclusions

This proposal was created as a first step toward the development of an integrated system that can provide assistance and support during catastrophic weather events, such as flooding. This idea was inspired by a desire to address the problem of flood inundation, which is becoming a more frequent severe event in modern times. In this light, we believe artificial intelligence

[5]the trained model was saved and can be found at the following link.https://github.com/andrearafanelli/Aerial-image-seg

[6]https://github.com/facebookresearch/detectron2/blob/main/README.md

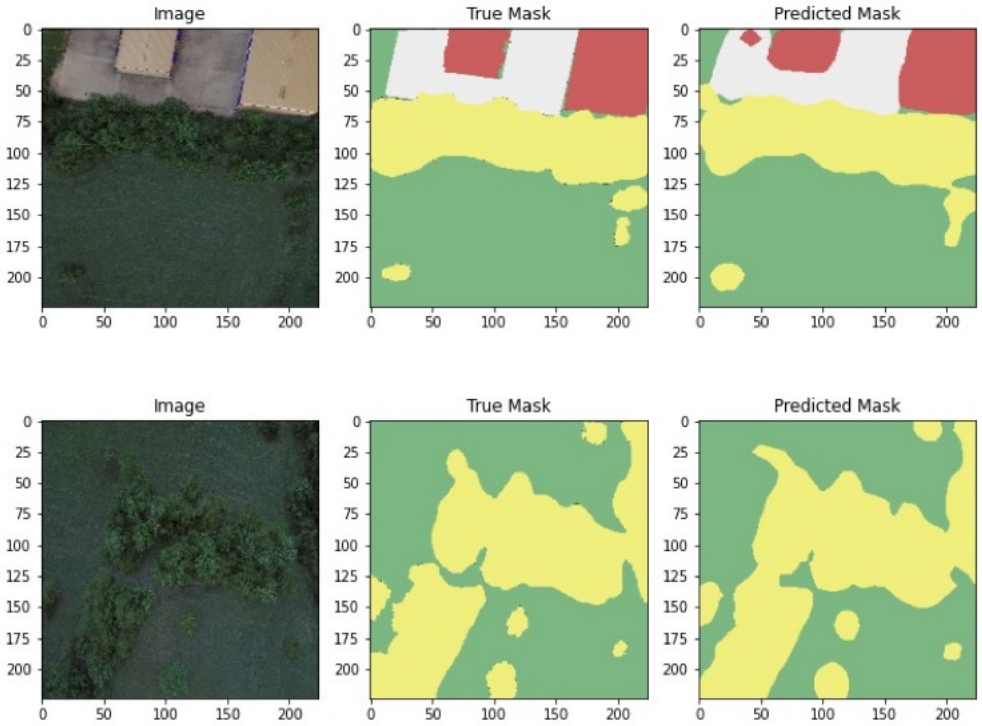

Figure 6: PSP-Net output examples

at large could provide a swift and effective answer for assisting people to manage disasters of this magnitude. We believe that the effectiveness of the combination of multi-agent paradigms and Machine Learning (ML) may be the main strength of this proposal. Undoubtedly, designing agents so that they can collaborate synergically and effectively is critical. This requires the identification of specific tasks for each agent, as well as the abstraction and formalization of the organizational structure of the multi-agent system. In the future, we intend to thoroughly develop each element of the proposed system, particularly the phase of translating the image into logical rules and the phase of reasoning. For data retrieval, we would also like to interface the system with external sources via APIs.

References

[1] S. H. Chen, A. J. Jakeman, J. P. Norton, Artificial intelligence techniques: An introduction to their use for modelling environmental systems, Mathematics and Computers in Simulation (MATCOM) 78 (2008) 379–400. URL: https://EconPapers.repec.org/RePEc:eee:matcom:v:78:y:2008:i:2:p:379-400.

[2] S. Costantini, A. Tocchio, The dali logic programming agent-oriented language, in: J. J. Alferes, J. Leite (Eds.), Logics in Artificial Intelligence, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004, pp. 685–688.

[3] S. Costantini, A. Tocchio, A logic programming language for multi-agent systems, in: S. Flesca, S. Greco, G. Ianni, N. Leone (Eds.), Logics in Artificial Intelligence, Springer Berlin Heidelberg, Berlin, Heidelberg, 2002, pp. 1–13.

[4] S. Costantini, A. Tocchio, About declarative semantics of logic-based agent languages, in: M. Baldoni, U. Endriss, A. Omicini, P. Torroni (Eds.), Declarative Agent Languages and Technologies III, Third International Workshop, DALT 2005, Selected and Revised Papers, volume 3904 of *Lecture Notes in Computer Science*, Springer, 2005, pp. 106–123.

[5] S. Costantini, G. De Gasperis, V. Pitoni, A. Salutari, Dali: A multi agent system framework for the web, cognitive robotic and complex event processing., in: ICTCS/CILC, 2017, pp. 286–300.

[6] H. Zhao, J. Shi, X. Qi, X. Wang, J. Jia, Pyramid scene parsing network, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[7] L. Mitchener, D. Tuckey, M. Crosby, A. Russo, Detect, understand, act: A neuro-symbolic hierarchical reinforcement learning framework, Mach. Learn. 111 (2022) 1523–1549. URL: https://doi.org/10.1007/s10994-022-06142-7. doi:10.1007/s10994-022-06142-7.

[8] M. Rahnemoonfar, T. Chowdhury, A. Sarkar, D. Varshney, M. Yari, R. Murphy, Floodnet: A high resolution aerial imagery dataset for post flood scene understanding, IEEE Access 9 (2021) 89644–89654.