Article

Explainable Hopfield Neural Networks by Using an Automatic Video Generation System

Clemente Rubio Manzano [1] 0000-0001-5832-070X

[1] Department of Information Systems, University of the Bío-Bío and Department of Mathematics, University of Cádiz; clrubio@ubiobio.cl
* Correspondence: clrubio@ubiobio.cl

Abstract: Hopfield Neural Networks (HNNs) are recurrent neural networks used to implement associative memory. Their main feature is their ability to pattern recognition, optimization, or image segmentation. However, sometimes it is not easy to provide the users with good explanations about the results obtained with them due to mainly the large number of changes in the state of neurons (and their weights) produced during a problem of machine learning. There are currently limited techniques to visualize, verbalize, or abstract HNNs. This paper outlines how we can construct automatic video generation systems to explain their execution. This work constitutes a novel approach to get explainable artificial intelligence systems in general and HNNs in particular building on the theory of data-to-text systems and software visualization approaches. We present a complete methodology to build these kinds of systems. Software architecture is also designed, implemented, and tested. Technical details about the implementation are also detailed and explained. Finally, we apply our approach for creating a complete explainer video about the execution of HNNs on a small recognition problem.

Keywords: Explainable Artificial Intelligence, Hopfield Neural Networks, Automatic Video Generation, Data-to-text systems, Software Visualization

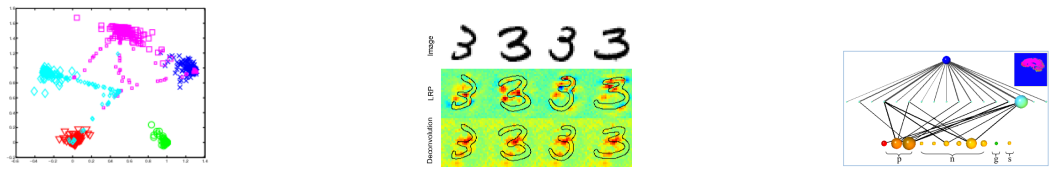
1. Introduction: AI and Explainability

Nowadays, transparency is one of the most critical words around the world [19,20]. We perceive it as the quality of easily seeing or understanding the others' actions, implying openness, communication, and accountability [7]. In computer science, transparency has been historically a significant challenge, and it has been addressed in different ways by different disciplines. We can mention here algorithmic transparency, algorithmic accountably [12,15,27,28,50] and, more recently, interpretable/explainable systems [23]. The common goal of all the above disciplines is that algorithms' actions must be easily understood by users (expert and non-experts) when we execute them in a particular context.

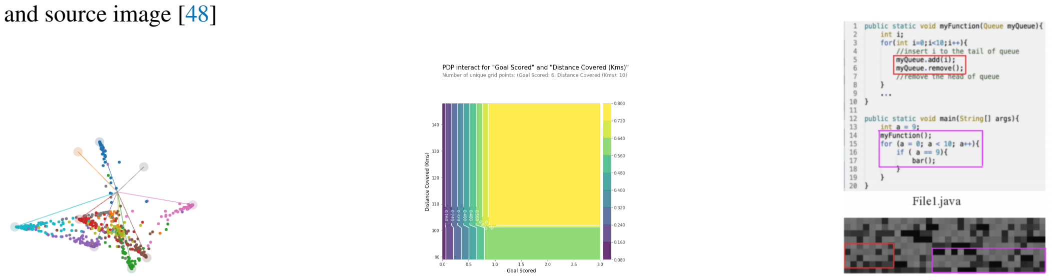
On the other hand, Artificial Intelligence (AI) is impacting on our everyday lives, improving decision-making and performing intelligent tasks (image and speech recognition, image and speech generation, medical diagnostic systems and more). However, despite AI programs' capabilities for problem-solving, they lack explainability [13]. For example, deep learning algorithms use complex networks and mathematics, which are hard for human users to understand. AI must be accessible for all human users not only for experts ones. Even, it could be a severe problem when unexpected decision need to be clarified in critical contexts: medical domain/health-care, judicial systems, banking/financial domain, bio-informatics, automobile industry, marketing, election campaigns, precision agriculture, military expert systems, security systems and education [6]. Therefore, it is a current challenge to understand how and why the machines make decisions. Even the European Union has fostered a framework of regulations for providing the scientific community with guidelines to get satisfactory explanations from the execution of algorithms governing decisions' making. [22].

On this regard, a current challenge in AI is to achieve explainability in AI [23]. In the literature, several computational methods have been proposed in order to explain predictions from supervised models [3,39,44]. On the other hand, there exists works whose objetive is to propose visualization and animation techniques. We find different kinds of visualization and animation methods, the Fig. 1 summarizes some of them. We can mention here the following:

1. Projections of network outputs to visualize decisions in a K-dimensional space. This approach aims to present a set of images for all training vectors [16].



(a) Proyections in K-dimensional space [16], Heatmaps [43], Dual representation with colored neural networks and source image [48]



```
public static void myFunction(Queue myQueue){
    int i;
    for(int i=0;i<10;i++){
        //insert i to the tail of queue
        myQueue.add(i);
        myQueue.remove();
        //remove the head of queue
    }
    ...
}

public static void main(String[] args){
    int a = 9;
    myFunction();
    for (a = 0; a < 10; a++){
        if ( a == 9){
            bar();
        }
    }
}
```

(b) Grand tour visualization technique[30], internal matrix of weights [41], code as an image [8]

Figure 1. Examples of graphical representations of different neural networks.

2. Saliency maps (or heatmaps) to simplify and/or change the visual representation of an picture into something that is more easier and meaningful to analyze [33,43].
3. Colored neural networks to show the changes in the states of neurons [48].
4. Visualization of programs as images by using a self-attention mechanism for extracting image features [8].
5. Animations by using the Grand Tour technique [30] in which animations are employed to have advantages with respect to classical visualization techniques [4,49].

Despite the facilities shown by the graphical tools, sometimes it is not easy to interpret them, especially when non-expert users are in charge of performing this task. For this reason, Interactive Natural Language Technology for Explainable Artificial Intelligence has been recently proposed [1]: *"the final goal is to make AI self-explaining and thus contribute to translating knowledge into products and services for economic and social benefit, with the support of Explainable AI systems. Moreover, our focus is on the automatic generation of interactive explanations in NL, the preferred modality among humans, with visualization as a complementary modality [...]"*. This paper follows this line of work.

On the other hand, NL is not the only, nor always, most efficient way to have humans communicate with computers. The use of video images in the context of an expert system was shown us that it is not enough to stick pictures to make the communication efficient and attractive [5]. The combination of graphics and text is an effective explanation strategy [40] because the users can focus their attention on the explanatory information in the text [32]. At the same time, graphics allows users to form mental models of the information explained in the text. The combination of visual and text information is a useful explanation strategy when it meets four conditions [31]: 1) the text is easy for all users to understand; 2) the visuals are created and evaluated based on the user's comprehension; 3) the visual representations are employed to provide users with good explanations of the texts; and 4) the experience of users with the content is little or no previous. Additionally, video is a ubiquitous data type not only for viewing (reading) but for daily communication and composition (writing) [11].

For all the above considerations, we propose Automatic Video Generation (AVG) as a new interactive NL Technology for Explainable AI whose objective is to automatically generate step-by-step explainer videos to improve our understanding of complex phenomena dealing with a large amount of information, HNNs in our case.

The idea is to combine in the same framework explanations in NL with visual representations of the data structures and algorithms employed in a process of machine learning and put them together in a sequence of frames (or slides). The result is a video presentation similar to those created by teachers, experts, or researchers when they need to explain something to an audience. Our approach is strongly context-dependent; hence, we will use this feature to provide users with

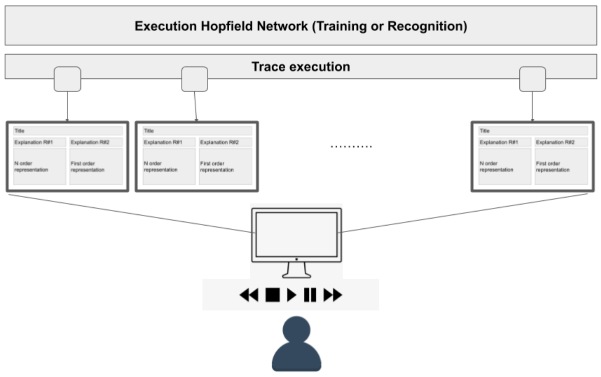


Figure 2. A general framework for transforming the execution of an algorithm in an explainer video by using data-to-text and software visualization techniques.

factual and realistic explanations. Moreover, storytelling techniques help us to include global commentaries. Additionally, as we have videos (mp4, slides, ppt) as output, the user can classically analyse them: forward, backward, pause, slow, fast, so on. Another significant advantage is that a complete understanding of the models could require several examples. Our approach can generate several videos from different samples quickly using a small number of changes. As this is a first approximation, we have some limitations; we highlight two: I) the difficulty of generating videos with large neural networks; ii) the limited degree of sophistication of the sentences generated in language. These limitations do not diminish the fact that our proposal is promising, and both limitations will address in future works.

Summarizing, the most relevant contributions of this paper are:

- We propose a new technology to automatically generate explainer videos from HNNs execution traces whose objective is to understand this process better.
- Our approach provides users with visual and textual explanations using data-to-text and software visualization techniques.
- We explain in detail a methodology and propose a software architecture.
- Technical details about the implementation are given and source code is provided on an official web page [1].
- We present an application to generate an explainer video on a real pattern recognition problem.

The rest of the paper is organized as follows. Section 2 introduces several preliminaries concepts about Hopfield neural networks and provides reader with a very brief review of the state of art on the different involved areas. Then, in section 3 a methodology and a software architecture based on CTP for the automatic generation of explainer videos are presented in detail. In section 4 are explained the most important task, namely, the content determination of frames (how the information will be delivered to the user) and the frames planning (in which order the information will be delivered to the user). Section 5 explains how narration scripts can be used to support the visual and textual explanations during the explainer videos' visualisation. In section 6 technical details about the implementation are given, the main algorithms are presented and explained. Finally, section 7 provides some concluding remarks and presents the future work with special attention in showing how and why we can use AVG as a powerful teaching-learning resource.

2. Preliminary concepts

2.1. Hopfield networks

HNNs are an essential type of recurrent artificial neural networks used to implement associative memory. Their value lies in their ability to pattern recognition, optimization or image segmentation. [9]. HNN's origin is the Ising model, a mathematical model employed in the well-known statistical mechanics to explain the ferromagnetism phenomenon. This model use variables to represent magnetic dipole moments of atomic spins. Two states are possible (+1 or 1). This feature is

[1] youractionsdefineyou.com/videogeneratorhopfield/

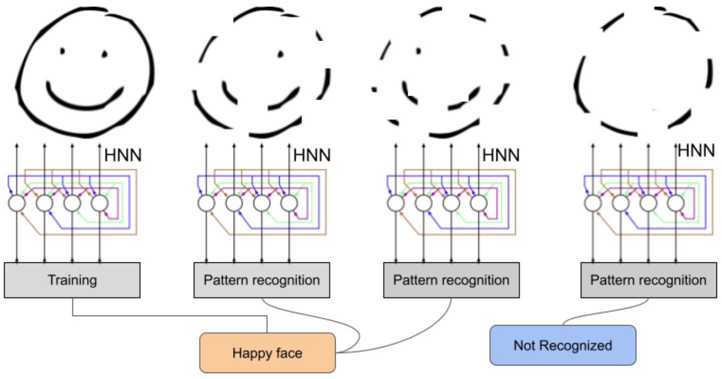


Figure 3. HNN applied to happy face recognition: training and pattern recognition phases.

employed in the Hopfield neural networks where each neuron only take values on two different states (+1 or 1) depending on the input value and if it exceeds their threshold U_i. In particular, Discrete Hopfield neural networks establish a relationship between each pair of neurons (i, j) with $1, 2, \ldots i, j, \ldots N$ [26].

In a particular instant of time, a vector V of N bits is employed to represent the state of the network. Each vector V stores information about which neurons are firing. The interactions between neurons are represented by using weights w_{ij} whose values are usually 1 or -1 [2]. In this kind of neural networks the Hebb's law of association is employed to learn these interactions, a neuron can not be connected with itself and the connections between neurons are symmetric. More formally, for a certain state V^s, a weight (i, j) is defined as follows:

$$w_{ij} = (2V_i^s - 1)(2V_j^s - 1)$$

$w_{ii} = 0, \forall i$ and $w_{ij} = w_{ji}, \forall i, j w_{ij} = w_{ji}, \forall i, j$. Then, the values for each vector are computed as follows:

1. $V_i \to 1$ if $\sum_j w_{ij} V_j > U_i$
2. $V_i \to 0$ if $\sum_j w_{ij} V_j < U_i$

The main feature of HNNs is their capability of remembering states stored in the vectors (interaction matrix). That is, each neuron is going to change until it matches the original state V^s if a new state $V^{s'}$ is subjected to the interaction matrix.

Example 1. *Suppose we want to use an HNN to recognize a happy face in a picture. The first step is to provide the system with a representative example. Once the system learns this pattern, it can recognize any example similar to it. The Figure 3 shows the three options that can occur: a) training an HNN with an example; b) recognizing two noise examples and ; c) when the HNN is not capable of identifying an example given.*

2.2. Data-to-text

Data-to-text systems (D2T) have been gaining relevance in AI during the last years. These systems are working in real-life applications (Galiweather [35] or Metoffice [37]) obtaining very promising results [42,45]. We are referring here to D2T as a discipline that includes three areas: Theory of Computational Perceptions (CTP), Linguistic Descriptions of Complex Phenomena (LDCP) and Natural Language Generation (NLG). CTP [53,54] is based on the way human beings use NL to reason, make decisions and communicate their experience in an context. The main feature of CTP approaches is that they can work in environments in which there exist imprecision, uncertainty and partial truth. CTP aims to develop intelligent computational systems capable of computing and reasoning with imprecise descriptions in NL in a similar way as humans do. On the other hand, the field of the Linguistic Description of Complex Phenomena (LDCP) theory [46] aims to automatically generate linguistic reports by using a conceptual framework based on CTP for the automatic

[2] We employ this convention in this paper. Other literature employ neurons that take values of 0 and 1.

generation of reports using natural language about a well-defined phenomenon. There are several real-life applications based on the LDCP framework, such as [10,17,36]. Finally, NLG systems [37] aims to transform databases in reports in natural language. The main difference between classical NLG systems and fuzzy approaches is the management and treatment of imprecision/vagueness. The first one allows us to obtain more elaborated reports in NL and the treatment of vaguenees need to be incorporparted in the system. In the second one, the treatment of vagueness is in the core, but the style of the linguistic reports is less rich. Both proposals have been evolving in a parallel way and they are complementary rather than competitive.

2.3. Film Generation, Software Visualization and 2D programming

Like Natural Language Generation, Film Generation aims to better comprehend a particular domain by automatically generating a film. Ethnologists [21] have identified at least two levels of film comprehension. The first level is concerned with understanding single images (determining what is in the picture, differentiation of objects in them, confusion between images and reality). The second one concerns the comprehension of the storyline. Film generation is related to software visualization (SV). SV is a visual computational technique usually employed in analysis and development of programs, software maintenance, reverse engineering, and collaborative development. SV aims to propose graphical representations (static or dinamic) of data structures, algorithms or programs (code) to get a better understanding of them when they are dealing with large amount of information. The main challenge of this discipline is to get good graphical representations using visual metaphors to explain different aspect of the software development. To implement these graphical representations, we need to use a library for 2D programming. We are going to use the Java Foundation Classes (JFC) which is a standard API for providing programmers with a graphical user interface (GUI) for Java programs. In particular, we employ all the functionalities of the well-known Abstract Window Toolkit (AWT). AWT is a library which allows programmers to graphically draw simple or complex representations in two dimensions by using a set of graphical primitives. In this paper, we employ the following primitives [3]:

- `drawString(string,x,y);` allows us to draw on the screen a string in the position (x, y).
- `drawRect(x,y,width,height);` allows us to draw on the screen a rectangle in the position (x, y) with a particular width and height.
- `fillRect(x,y,width,height);` allows us to draw on the screen a filled rectangle in the position (x, y) with a particular width and height. The color of the fill can be modified by the following function.
- `setColor(Color);` allows us to change the current color employed in the current privitive. Colors can be changed any time, any where.
- `setFont(Font);` allows us to change the font employed in the current `drawString` privitive. Fonts can be changed any time, any where.

A useful feature of our approach is that more complex graphical represenations can be created by combining these simple primitives (see Section 6 and the function called `drawPattern` for more detail). Additionally, we will use the graphviz software which is as the authors said: *"open source graph visualization software. Graph visualization is a way of representing structural information as diagrams of abstract graphs and networks. It has important applications in networking, bioinformatics, software engineering, database and web design, machine learning, and in visual interfaces for other technical domains."* [18].

3. Automatic explainer video generation

In this section, we introduce the philosophy behind automatic explainer video generation. Additionally, we focus on two crucial aspects: i) how to design an AVG system based on CTP and software visualization and; ii) how should be implemented these kinds of systems. We propose a methodology to answer to item i), and software architecture to answer the item ii).

[3] Note that, others primitives could be used to enrich the graphical representations of our approach, it will depend on the available time for designing and implementation of each AVG system

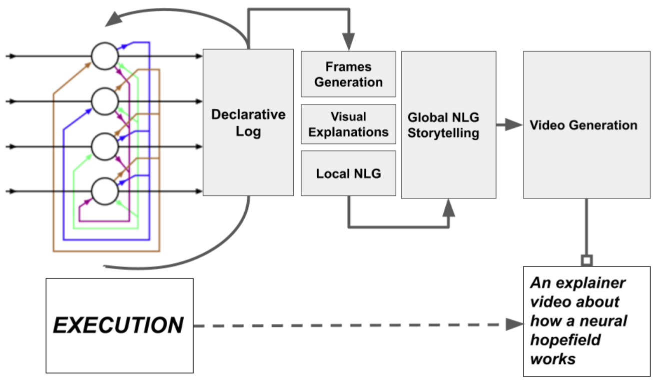


Figure 4. A general methodology for transforming hopfield networks execution in an explainer video by using four phases: trace, frames generation/local NLG, storytelling/global NLG and video generation.

3.1. The design of video generation system based on CTP

Our design is based on two crucial aspects: experience and human-computer interface which were pointed out in the work [47].

1. **Experience.** In this context, the experience [24] can be defined as the descriptions employed to explain a complex phenomenon which is formed by two levels of reality. The first level (also called first order phenomena) is directly related with the environment. The second one (also called second order phenomena) is formed by the meanings and wordings of the perceptions of the first order one. The desired result is to get AI systems (personal assistans) capable of helping us to automatically create our personal experience. For defining the experience in HNNs, we will employ first-order representations related directly to the domain, that is, algorithms and data structures used to implement them. And the second-order ones formed by graphical models and abstractions of the first ones to facilitate their understanding. Hence, our idea is to get an artificial teacher assistant capable of explaining the execution of HNNs and provide users with good explanations about the process of training and patten recognition.
2. **Human-Computer Interface.** We must consider three fundamental items when our objective is to communicate meaning with NL by using an intelligent and automatic personal assistant system [24]. First, we need to establish a relation between the users and the AI system. Second, we have to define the domain of language between them. Third, a mode of expression must be defined (informative, persuasive, didactical, etc). Additional features could be taken into account. For example, the use of graphics and sounds could be considered in order to complement explanations in NL or; the capability of expressing emotions with the tone of voice and the facial expressions of an avatar. Our human-computer interface is an explainer video formed by the following elements: i) concepts related to HNNs (matrix for representing the activation of neurons, graph for illustrating the connections between neurons, so on); ii) a student-teacher relationship; iii) the mode of expression will be a set of frames formed by explanations in NL and complementing it with the use of graphics. This video presentation aims to be informative and didactical.

The design and the implementation of our human-computer interface will perform by using a combination of two computational disciplines, namely: data-to-text systems (CTP, LDCP and NLG) and software visualization by using 2D programming. Each one of them is dedicated to performing a particular task. In the following, we clarify the tasks performed by each component.

- **Data-to-text (CTP + LDCP + NLG).** Firstly, we employ CTP as a tool for knowledge representation capable of capturing a human expert's perceptions, in this case, a teacher whose objective is to teach and explain HNNs. As we mentioned, our philosophy is creating explainer videos in a similar way than a human teacher creates them by using a set of slides formed by images and explanations in NL. Secondly, we use LDCP to define two kinds of computational perceptions, but now we will call them graphical representations (first and second order representations). The first ones will represent internal structures as variables or data structures. The

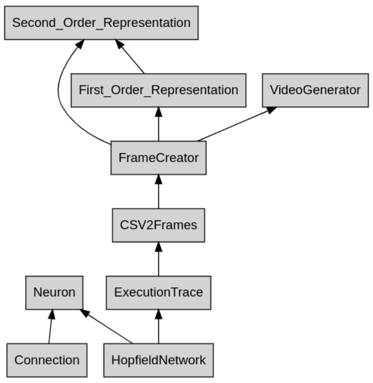


Figure 5. Diagram about the relations between the different concepts and their relations.

second order ones represent more abstract graphical representations and we define them from the first-order ones in a similar way than we calculate n order perceptions from first-order ones in LDCP. Finally, we employ NLG to generate pieces of sentences in natural language. We adapt the techniques investigated in this discipline to create sentences in NL from graphical representations. Also, we establish the content determination and planning phases to work with pictures and sentences.

- **Film Generation by using software visualization and 2D Programming.** We start with a representation of shots frames. The main challenge is finding good graphical representations using visual metaphors and NL explanations. We aim to combine both communication mechanisms by creating visual and textual structures. This technique is employed to generate 2D graphical representations by using 2D programming. We will transform the symbolic terms captured in the execution trace in a set of 2D visual drawings which will incorporate into the process of video generation.

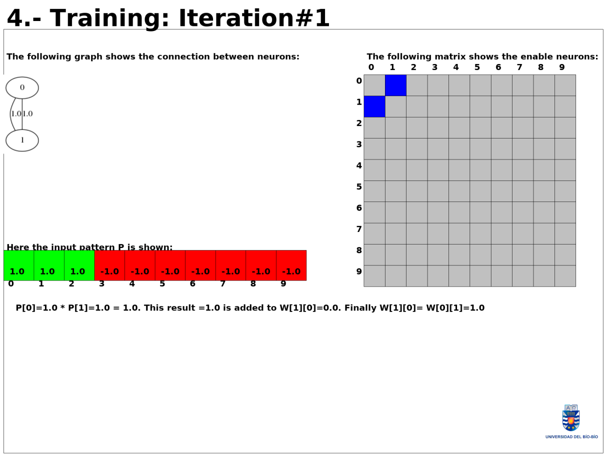
3.2. An software architecture to build AVG based on CTP

We propose a software architecture based on the previously presented methodology. Four modules form the architecture: declarative log, frame generation/local NLG, global NLG/storytelling and video generation.

1. **Declarative log.** As it is well-known, traceability of neural network training is a very complex task because it employs so many neurons, connections and weights what it implies so many iterations. This module aims to capture the values of the data structures generated during an HNN execution and generate a declarative log from them that the next modules can use. Information about neurons, connections and weights will store by using an ontology [34]. The reason for using a declarative log is double: first, we have a formal knowledge representation of the execution process, and second, it can help users to interpret the captured in the next phases, for example, when the system perform the content determination and planning phases. More formally, we define a symbolic video frame by using a fact with the following form `frame(id, action,[data_structures]);` where `action` indicates the action performed in that instant of time identified by `id` and `[data_structures]` are the values of each data structure associate to a particular action. Therefore, the complete execution of an HNN is as follows:

```
frame(1,action_1,[data_a1]).
frame(2,action_2,[data_a2]).
...
frame(n,action_n,[data_an]).
```

   We explain the transformation of this set of symbolic frames to visual ones in section 4. Additionally, we describe technical details about the implementation in section 6.
2. **Frames generation and local NLG.** This paper adopts the classical architecture of NLG systems, but we adapt it for automatic frames generation. An essential feature of our approach



4.- Training: Iteration#1

The following graph shows the connection between neurons:

The following matrix shows the enable neurons:

	0	1	2	3	4	5	6	7	8	9
0	⬜	🟦	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜
1	🟦	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜
2	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜
3	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜
4	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜
5	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜
6	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜
7	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜
8	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜
9	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜

Here the input pattern P is shown:

1.0	1.0	1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0
0	1	2	3	4	5	6	7	8	9

P[0]=1.0 * P[1]=1.0 = 1.0. This result =1.0 is added to W[1][0]=0.0. Finally W[1][0]= W[0][1]=1.0

Figure 6. Frame automatically generated in which textual explanations are combined with graphical representations (first-order and second-order).

is that the system generates NL descriptions from internal representations of variables and data structures. It consists of four steps: 1) the content determination subsystem computes an explanatory model of the result of the execution of HNNs. It employs two kinds of visual explanations: first-order representations for visually representing internal variables and data-structures; and second-order ones for more abstract models of them. The system creates second-order ones from the first ones. It is fed to a planning system and identifies which parts of these representations must be explained and in what order.

3. **Global NLG - Storytelling.** This module implements the “voice" of the expert or teacher. The aim is to find the perfect combination of narrative and data. The idea is to employ storytelling as it is used in others disciplines as training and education contexts : to motivate and to illustrate. The standard approach to incorporating storytelling into a computational intelligent system are the well-known scripts. We employ a script based on classical presentations, that is, we define the first frame with a little explanation about the presentation’s context. Next, we put an index about the main items that we will explain during the video. Finally, we define a slide for the last remarks. On the other hand, we need to be capable of defining extra explainer slides on groups of local slides. This process can be seen as a global NLG system which takes as input a set frames (texts and visual explanations) and automatically create slides. Each extra slide is putting before them with an explanation about that group of slides. For example, an extra explainer slide for the training phase is the one created from the explanations of a group of slides generated to explain the training process. The same process could be perfomred for the pattern recognition phase (see section 5 and Figure 8).
4. **Video Generation.** This module aims to generate a video from a set of frames automatically. In this work, we do not address the automatic selection of the type of letters, colours, discourse, etc.

4. Frames Generation and local NLG

4.1. Content determination of frames

In this section, we will explain the content determination of the frames. Content determination identifies the information we want to communicate through the generated explainer video. Therefore, video frames will be a set of explanations in NL and graphical representations. The central concept here is the neural network. To define a neural network data type, we need to define two types of data: a set of neurons and a set of connections between them. Therefore, each frame’s content provides users with information about these types’ values. We employ here the UML language for defining the data types and their relations.

In particular, the Figure 5 shows the different data types and their relations (Neuron, Connection, Neural Networks). Each of them contains fields (attributes or properties), and code in procedures (often known as methods). We represent and store the information generated during the HNN execution in a declarative log (ontology). We define each fact "frame" as a tuple of four elements: a unique identifier that indicates the order of execution; a label action gives us information about which operation was performed (initialization, neurons, connections, iteration, so on); information about the values of the data structures employed by the HNN and in some cases a set of messanges. Different kind of messages can be generated from each frame. For example, the execution of an HNN can generate a declarative log of this kind (we omit some details for simplicity):

```
frame(1,neurons,[N0,N1,N2,N3,...,N6,N7,N8,N9],
                 "The neurons defined for the
                 HopField network are:").

frame(2,connections,[(N0,N1,0.0),...,
                                 (N9,N9,0.0)],
                "The connections between
                neurons have been
                established:").

frame(3,pattern([1.0,1.0,1.0,-1.0,-1.0,
                -1.0,-1.0,-1.0,-1.0,-1.0],
                "The input pattern is:")).

frame(4,iteration#1,[w(0,1,1.0)],
            "Neuron N0 is enable.
             The neuron N0 is
             connected with
             neuron N1 and N2").

frame(5,iteration#2,[w(0,2,1.0),
                    (w(1,2,1.0)]).
...
frame(13,iteration#9,[(w,0,9,-1.0), ...
                      (w,8,9,1.0)]).
```

Additionally, we employ graphical representations to provide users with useful and visual explanations about the changes produced in the internal states of an HNN during the execution. These representations will be essential in the next phases to get simple, informative and intuitive messages about the neurons, connections, neural network and training/pattern recognition tasks. We define two types of visual explanations, namely: first-order representations to visually explain graphical low-level data structures, variables or values involved in the process of execution; and the second-order ones to model more abstract figures. An essential feature of our approach is that the system generates second-order ones from the first ones automatically in a similar way than computational perceptions are created in the LDCP paradigm. The Figure 6 shows both types of representations. We represent neurons, connections, and weights as a matrix of weights on the right side. On the left side, a classical graph represents the neurons' activation, vertices are the neurons, edges are the connections, and the labels are the weights. Each visual explanations will be associated with a set of messagess explaining the pieces of information shown in them.

4.2. Frames planning

In essence, a plan is a sequence of actions that bring a system from an initial state to a goal state. A common approach to AI planning involves defining a series of atomic steps to achieve a communication goal. When the communication goal is to explain an execution of HNNs, then the plan is partially guided by the implementation and the execution trace. This last one provides the planner with a sequence ordered of frames. In our case, the steps needed to explain the HNN execution it is as follows (see Figure 7):

1. **Initialization**. The system (using content determination resources) visually show and textually explain the neurons, connections, neural networks weights and input pattern.

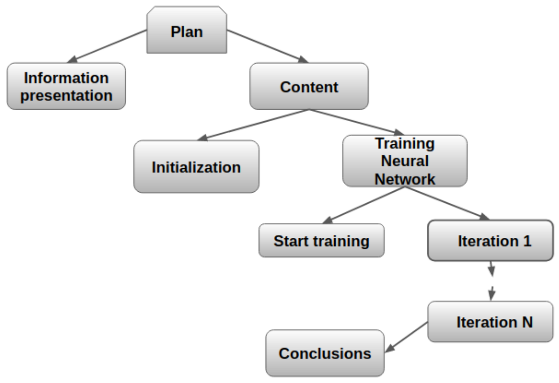


Figure 7. Planing for the video presentation of HNNs execution (training).

2. **Training**. The system (using content determination resources) visually show and textually explain the training procedure step by step. The system can employ a set of frames in order to provide user with useful information about it.
3. **Pattern Recognition**. The system (using content determination resources) visually show and textually explain the pattern recognition procedure step by step. The system can employ a set of frames in order to provide user with useful information about it.

Additionally, as the system using a particular plan to simulate a teacher's presentation, it incorporates some extra frames before explaining the training and pattern recognition tasks. We add three slides more to the original sequence: Initial presentation (title, contact, date, so on); an index with little explanations of each item and; a slide (or several) for conclusions.

5. Global NLG - Storytelling and narrative

This section explains how to employ a narration script for supporting the visual and textual explanations during the explainer videos' visualisation. A script is a standard approach to incorporating storytelling into a computer system. In [2] some recommendations are given to get a good narrative in the presentations:

- **Good examples.** The use of illustrative and specific examples to show your ideas and explain the scope of the talk in the time available.
- **Detective story.** The narrative style must follows a detective story structure. The speaker begins by presenting the specific problem and then he/she describes the search for a solution.

We aim to find the perfect combination of data and narrative. We use a narrative style similar to the one used in training and education contexts to illustrate and to motivate. Our storytelling module is the "voice" of the artificial expert or teacher. In our case, we employ a template-based approach. We focus on the logical causal progression of representations, as it is mentioned in [38]: *"Causality refers to the notion that there is a relationship between temporally ordered events. One event changes the story world in a particular way that enables future events to occur"*. In our case, a story expresses essentially how and why an HNN changes. The HNN storytelling consists of explaining the current changes produced during the execution, taking into account the past ones and establishing some relations (possibly casual) between them.

In its current version, our template script is straightforward. We establish a video script in two steps. First, we define each group of the slides from the items defined in the index slide. Second, we create an explainer slide for and from a group of slides. This special slide aims to summarize all the explanations generated to explain a complex process (for example, training or pattern recognition). We use here a first prototype of an avatar for simulating a human teacher. The storytelling is optional; the users can choose if they want to extra explanations about some slides. These slides could remove by the user or using as support notes. Figure 8 summarizes this process.

6. Technical details about the implementation

In this section, technical details about the implementation are given. The first implementation has been developed in Java 11. Several explainer videos and the source code can be watched and

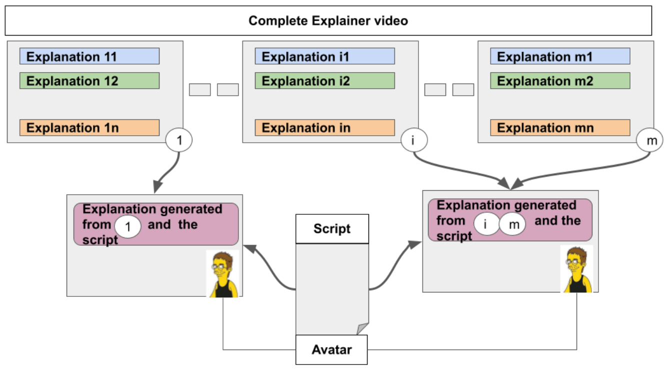


Figure 8. Storytelling pipeline.

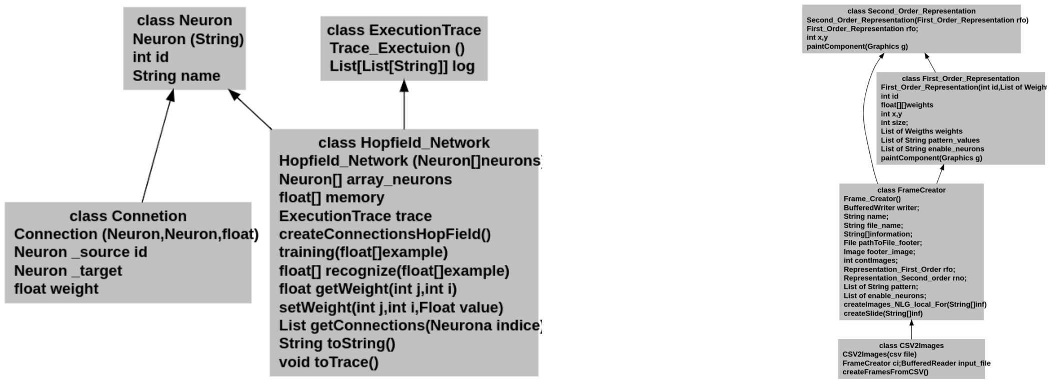

class Neuron
Neuron (String)
int id
String name
class ExecutionTrace
Trace_Exectuion ()
List[List[String]] log
class Connetion
Connection (Neuron,Neuron,float)
Neuron _source id
Neuron _target
float weight
class Hopfield_Network
Hopfield_Network (Neuron[]neurons
Neuron[] array_neurons
float[] memory
ExecutionTrace trace
createConnectionsHopField()
training(float[]example)
float[] recognize(float[]example)
float getWeight(int j,int i)
setWeight(int j,int i,Float value)
List getConnections(Neurona indice
String toString()
void toTrace()
class Second_Order_Representation
Second_Order_Representation(First_Order_Representation rfo)
First_Order_Representation rfo;
int x,y
paintComponent(Graphics g)
class First_Order_Representation
First_Order_Representation(int id,List of Weigh
int id
float[][]weights
int x,y
int size;
List of Weigths weights
List of String pattern_values
List of String enable_neurons
paintComponent(Graphics g)
class FrameCreator
Frame_Creator()
BufferedWriter writer;
String name;
String file_name;
String[]information;
File pathToFile_footer;
Image footer_image;
int contImages;
Representation_First_Order rfo;
Representation_Second_order rno;
List of String pattern;
List of enable_neurons;
createImages_NLG_local_For(String[]inf)
createSlide(String[]inf)
class CSV2Images
CSV2Images(csv file)
FrameCreator ci;BufferedReader input_file
createFramesFromCSV()

Figure 9. The diagram of classes and their main elements (attributes and methods of each class).

downloaded from the official web page of this paper [4]. The complete implemention is summarized by using UML diagrams (see Figure 9).

The implementation is formed by two packages: Hopfield neuronal network and video generation. The first one implements all the functionality required to execute the HNN (training and pattern recognition). The execution of an HNN consists of performing three steps: i) to receive as input an array of 1 and -1 (the pattern); ii) to train the HNN for recognizing this pattern; and iii) to recognize any array received as input. This package has a class called `ExecutionTrace` for storing all the information about the HNN execution. The result is a structured file containing a sequence of rows with information about the execution. Each row is formed by an identifier (`ID`) that indicates the order in which a row must be generated; a label action (`LABEL_ACTION`) gives us information about which operation was performed (initialization, creation neurons, creation connections, creation neural networks, weights update, so on). After that, a set of values `D1; ...; DN` is provided with information about the results obtained after the execution of the action indicated previously. Finally, a set of messages about each D_i could be also indicated in order to support the process of the video generation.

```
ID0;Label_Action;D1; D2;...;DN; M1;...;MN;
ID1;Label_Action;D1; D2;...;DM; M1;...;MM;
...
ID1;Label_Action;D1; D2;...;DL; M1;...;ML;
```

The video generation package implements an explainer video's automatic generation from the first package's execution trace. The main classes are CSV2Frame, FrameCreator, First-Order Representation, Second-Order Representation and VideoGenerator. The main algoritm 1 receives as input an execution trace (list of rows) and returns an explainer video in mp4. It reads one by one each line and creates a slide using the information associated with each row calling to the function `Create-Slide(row[])` (Alg 1-line 5).

[4] youractionsdefineyou.com/videogeneratorhopfield/

Algorithm 1: Video Generation (Execution Trace T) returns a video in mp4.

```
V: Set of Frames
begin
    while not(end_file) do
        Row[]=Read-Next-Line(T)
        V.insert(Create-Slide(row[]))
    end
    planning(V)
    story-telling(V)
    Vmp4=convert(V,mp4)
end
return Vmp4
```

The function `Create-Slide(row[])` creates an empty slide, and it will call a sequence of functions whose objective is to create, when correspond, a frame with visual and textual explanations. The mentioned functions are:

- `first-order-representations`
- `second-order-representations`
- `explanations-NL-for`

The first and second functions generate visual representations of the information received as parameters. The third function generates textual explanations from the visual ones. It is important to mention that sometimes we do not employ this function in the current implementation, but it will be an essential resource of explainability in the future. The algoritm 2 shows the pseudocode to create a new frame and the algoritm 3 shows how we implement the function `first-order-representations`. As we can observe this function creates a frame depending on the action received as parameter. For example, we explain here two kind of them: a simple one (`images-text-pattern(info)`); and a more complex one (`images-text-iteration(info)`) which is called complete cycle of represenation.

Algorithm 2: create-Slide(row-trace []).

```
begin
    //empty slide is created
    graphics = createFrameBackGround-Style()
    //graphical representations and texts
    //are generated
    rfo=first-order-representations(info)
    efo=explanations-NL-for(rfo)
    rso=second-order-representations(efo)
    eso=explanations-NL-for(rso)
    //graphical representations //are positioned
    updatePositions([rfo,efo,rso,eso])
    //graphical representations //are painted
    paint(graphics,[rfo,efo,rso,eso])
end
```

The function `images-text-pattern(info)` generates an explainer frame about the input pattern from an array of information `info[]` that contains the explanations generated to explain the data structures employed to implement this element. This function uses 2D atomic primitives to show the messages and create a graphical representation of the input pattern. The Figure 10 graphically explains this process. Sometimes atomic primitives are not enough to implement a particular graphical representation, and we need to create additional functions. For example, we define the function `drawPattern` that allows us to draw a visual matrix with different options: matrix alone, showing indexes or showing values. The possibilities will indicate in the parameters. If wn=true, then it creates a visual matrix showing their values; if wi=true, then it creates a graphical matrix with indexes (see Figure 10).

For the generation of a group of frames whose objective is to explain a particular phase's execution, the system must perform a complete cycle of representation. That is to say, for each frame, it generates a first-order model from the information provided by the neurons, connections and the

Algorithm 3: First-order-representation(info[]).

```
begin
    switch info[0] do
        case "initialization" do
            images-text-initialization(info)
        end
        case "neurons" do
            images-text-neurons(info)
        end
        case "connections" do
            images-text-connections(info)
        end
        case "pattern" do
            images-text-pattern(info)
        end
        case "iterations" do
            images-text-iterations(info)
        end
        case "conclusions" do
            images-text-conclusions(info)
        end
    end
end
```

Algorithm 4: images-text-pattern(info[]).

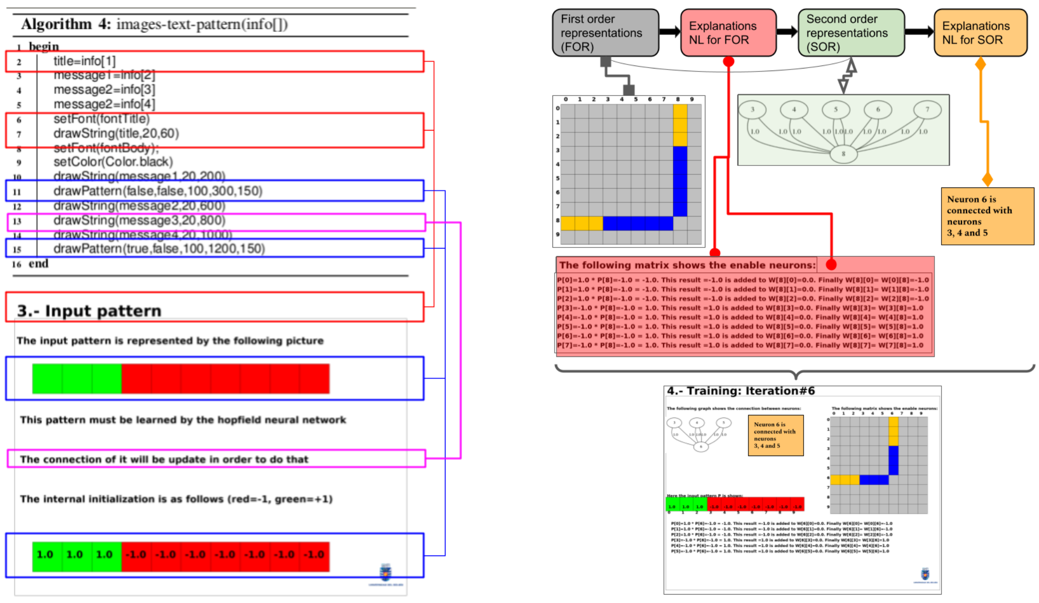
```
begin
    title=info[1]
    message1=info[2]
    message2=info[3]
    message2=info[4]
    setFont(fontTitle)
    drawString(title,20,60)
    setFont(fontBody);
    setColor(Color.black)
    drawString(message1,20,200)
    drawPattern(false,false,100,300,150)
    drawString(message2,20,600)
    drawString(message3,20,800)
    drawString(message4,20,1000)
    drawPattern(true,false,100,1200,150)
end
```

matrix of weights in a particular instant of time during the execution; and textual explanations about that. If the new representation needs to be clarified (it is a designer's decision), the system can generate additional descriptions. Next, the system generates a second-order model from the first one. It aims to provide users with an alternative and more friendly representation of the first one (the system can also generate explanations for this representation). The Figure 10 shows how this cycle is performed, and the concrete details about the implementation can be consulted in the source code on the official web page.

7. Conclusions and future work

A new framework based on data-to-text and software visualization to automatically generate explainer videos about the Hopfield neural networks' execution has been presented. A design based on experience and human-computer interface has been presented. A software architecture to build automatic video generation based on CTP has been explained in detail. Four modules form it:

1. Trace (for capturing the values of the data structures in each instant of time and other relevant information generated in the process of execution).
2. Frame generation/Local NLG (for creating slides containing texts and graphical representations about that).
3. Global NLG/Storytelling for the incorporation of texts which allows us to get a global communication.



```
Algorithm 4: images-text-pattern(info[])
begin
    title=info[1]
    message1=info[2]
    message2=info[3]
    message2=info[4]
    setFont(fontTitle)
    drawString(title,20,60)
    setFont(fontBody);
    setColor(Color.black)
    drawString(message1,20,200)
    drawPattern(false,false,100,300,150)
    drawString(message2,20,600)
    drawString(message3,20,800)
    drawString(message4,20,1000)
    drawPattern(true,false,100,1200,150)
end
```

3.- Input pattern
The input pattern is represented by the following picture
This pattern must be learned by the hopfield neural network
The connection of it will be update in order to do that
The internal initialization is as follows (red=-1, green=+1)
1.0 1.0 1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0
First order representations (FOR)
Explanations NL for FOR
Second order representations (SOR)
Explanations NL for SOR
Neuron 6 is connected with neurons 3, 4 and 5
The following matrix shows the enable neurons:
4.- Training: Iteration#6

```
P[0]=1.0 * P[8]=-1.0 = -1.0. This result =-1.0 is added to W[8][0]=0.0. Finally W[8][0]= W[0][8]=-1.0
P[1]=1.0 * P[8]=-1.0 = -1.0. This result =-1.0 is added to W[8][1]=0.0. Finally W[8][1]= W[1][8]=-1.0
P[2]=1.0 * P[8]=-1.0 = -1.0. This result =-1.0 is added to W[8][2]=0.0. Finally W[8][2]= W[2][8]=-1.0
P[3]=-1.0 * P[8]=-1.0 = 1.0. This result =1.0 is added to W[8][3]=0.0. Finally W[8][3]= W[3][8]=1.0
P[4]=-1.0 * P[8]=-1.0 = 1.0. This result =1.0 is added to W[8][4]=0.0. Finally W[8][4]= W[4][8]=1.0
P[5]=-1.0 * P[8]=-1.0 = 1.0. This result =1.0 is added to W[8][5]=0.0. Finally W[8][5]= W[5][8]=1.0
P[6]=-1.0 * P[8]=-1.0 = 1.0. This result =1.0 is added to W[8][6]=0.0. Finally W[8][6]= W[6][8]=1.0
P[7]=-1.0 * P[8]=-1.0 = 1.0. This result =1.0 is added to W[8][7]=0.0. Finally W[8][7]= W[7][8]=1.0
```

Figure 10. (Left) Frame generated by using the algortihms 3, 4 and 5; (Right(A complete cycle of representation for a frame extracted from a group of them whose objevtive is to explain the training phase)

4. Video generation (for generating a video in mp4 format from a set of images).

We have focused on explaining the content determination of frames and planning phases. Technical details about the implementation have been shown, the main algorithms and data structures have been explained. A real application for the explainer videos has been performed.

As future work we would like to work in the following challenges:

1. To improve the video by incorporating sounds, voices and more advanced avatars.
2. To create an automatic style module for providing users with options about visual aspects of the video (colours, font, speed, so on).
3. To investigate new methods for generating more rich sentences.
4. To be able to work with large neural networks by using intelligent analysis and automatic summarization of videos [25].
5. To employ links in which users can directly interact with the explainer video.

Special mention about future work is related to investigating how automatic explainer videos generation can be used as video tutorials for teaching-learning processes. The main reason is that videos online have become one of the most consumed digital resources, and millions of viewers watch them on youtube and other platforms. For example, a tutorial on skin retouching in photoshop has been watched for more than a million times and a youtube channel on photoshop tutorials has more than 170 thousand subscribers. In higher education, videos have also become an essential resource for students. They are integrated and employed as a fundamental part of traditional courses and they serve as a cornerstone of many courses. On the other hand, they have become in the primary information-delivery mechanism in online classes. In fact, several experts have pointed out that e-learning is the future of education. On this regard, Global Industry Analysts projected e-learning will grow in the future [29]. Also, the current pandemic has driven the use of video tutorials which play an important role in the e-learning area. At the same time, it has been shown how the use of presentation software instructions on video has advantages with respect to those performed on paper [51]. Currently, video tutorials (recorded or live) are a fundamental resource for students because they provide them with another perspective which becomes the experience of learning in a more dynamic, practical and effective experience [52]. For all these reasons, automatic explainer video generation can be a useful approach for digital and classical learning in the future.

Author Contributions: The paper has an only one author

Funding: University of the Bío-Bío

Algorithm 5: drawPattern(wn,wi,x,y,size)

```
begin
    xcell=x
    ycell=y
    size=s
    for $i = 0$ to pattern.length do
        if wi=true then
            drawString(""+i,xcell+15,y+size+20);
        end
        if patttern[i]=="-1.0" then
            setColor(Color.red)
        end
        else
            setColor(Color.green)
        end
        fillRect(xcell,ycell, size,size); setColor(Color.black); drawRect(xcell,ycell, size,size);
        if wn=true then
            drawString(pattern[i],xcell+20,ycell+80);
        end
        xcell+=size;
    end
end
```

Institutional Review Board Statement: Not apply

Informed Consent Statement: Not apply

Data Availability Statement: Not apply

Acknowledgments: SOMOS Research Group

Conflicts of Interest: The author has no conflict of interest

Sample Availability: Not apply

References

1. Alonso, J. M. et al. Interactive Natural Language Technology for Explainable Artificial Intelligence Workshop on Foundations of Trustworthy AI integrating Learning, Optimisation and Reasoning, at the European Conference on Artificial Intelligence
2. Anderson, C., and Duarte, N. (2013). How to give a killer presentation. Harvard business review, 91(6), 121-125.
3. Antwarg, L., Shapira, B., and Rokach, L. (2019). Explaining anomalies detected by autoencoders using SHAP. arXiv preprint arXiv:1903.02407.
4. Archambault, D., Purchase, H., and Pinaud, B. (2010). Animation, small multiples, and the effect of mental map preservation in dynamic graphs. IEEE transactions on visualization and computer graphics, 17(4), 539-552.
5. Bloch, G. R. (1988). From concepts to film sequences. In User-Oriented Content-Based Text and Image Handling (pp. 760-767).
6. Burkart, N., and Huber, M. F. (2020). A Survey on the Explainability of Supervised Machine Learning. Journal of Artificial Intelligence Research X, 1-74
7. Castelvecchi, D. (2016). Can we open the black box of AI?. Nature News, 538(7623), 20.
8. Chen, J., Hu, K., Yu, Y., Chen, Z., Xuan, Q., Liu, Y., and Filkov, V. (2020, June). Software visualization and deep transfer learning for effective software defect prediction. In Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering (pp. 578-589).
9. Cheng, K. S., Lin, J. S., and Mao, C. W. (1996). The application of competitive Hopfield neural network to medical image segmentation. IEEE transactions on medical imaging, 15(4), 560-567.
10. P. Conde-Clemente, J.M. Alonso and G. Trivino. Toward automatic generation of linguistic advice for saving energy at home Soft Computing, 1-15 (2016).
11. Davis, M. (1994, August). Knowledge representation for video. In AAAI (pp. 120-127).
12. Datta, A., Sen, S., and Zick, Y. (2016, May). Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems. In 2016 IEEE symposium on security and privacy (SP) (pp. 598-617). IEEE.
13. Darlington, K. (2017) Explainable AI Systems: Understanding the Decisions of the Machines In BBVA Open Mind (www.bbvaopenmind.com/en/technolog intelligence/explainable-ai-systems-understanding-the-decisions-of-the-machines/)
14. Dhaliwal, J. S. (1993). An experimental investigation of the use of explanations provided by knowledge-based systems. Doctoral dissertation, University of British Columbia).
15. Diakopoulos, N., and Koliska, M. (2017). Algorithmic transparency in the news media. Digital journalism, 5(7), 809-828.
16. Duch, W. (2003, July). Coloring black boxes: visualization of neural network decisions. In Proceedings of the International Joint Conference on Neural Networks, 2003. (Vol. 3, pp. 1735-1740). IEEE.

17. L. Eciolaza and M. Pereira-Fariña and G. Trivino Automatic linguistic reporting in driving simulation environments Applied Soft Computing 13(9), 3956 – 3967 (2013).
18. Ellson, J. et al. (2001, September). Graphviz—open source graph drawing tools. In International Symposium on Graph Drawing (pp. 483-484). Springer, Berlin, Heidelberg.
19. Fluck, M., and McCarthy, D. R. (2019). Information is power? Transparency and fetishism in international relations. Globalizations, 16(1), 1-16.
20. Florini, A. (2007). The right to know: transparency for an open world. Columbia University Press.
21. Forsdale, J. R., and Forsdale, L. (1966). Film literacy. Journal of the University Film Producers Association, 18(3), 9-27.
22. Goodman, B., and Flaxman, S. (2017). European Union regulations on algorithmic decision-making and a "right to explanation". AI magazine, 38(3), 50-57.
23. Gunning, D. (2017). Explainable artificial intelligence (xai). Defense Advanced Research Projects Agency (DARPA), nd Web, 2(2).
24. Halliday, M. A. K., and Matthiessen, C. (2006). Construing experience through meaning: A language-based approach to cognition. Bloomsbury Publishing.
25. He, L., Sanocki, E., Gupta, A., and Grudin, J. (1999, October). Auto-summarization of audio-video presentations. In Proceedings of the seventh ACM international conference on Multimedia (Part 1) (pp. 489-498).
26. Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. Proceedings of the national academy of sciences, 79(8), 2554-2558.
27. Kroll, J. A., Barocas, S., Felten, E. W., Reidenberg, J. R., Robinson, D. G., and Yu, H. (2016). Accountable algorithms. U. Pa. L. Rev., 165, 633.
28. Kemper, J., and Kolkman, D. (2019). Transparent to whom? No algorithmic accountability without a critical audience. Information, Communication & Society, 22(14), 2081-2096.
29. MCCUE, T. E learning climbing to 325 billion dollars by 2025 uf canvas absorb schoology moodle, 2018.
30. Li, et al. Visualizing Neural Networks with the Grand Tour Distill, 2020.
31. Mayer, R. E., and Gallini, J. K. (1990). When is an illustration worth ten thousand words?. Journal of educational psychology, 82(4), 715.
32. Mayer, R. E. (1989). Systematic thinking fostered by illustrations in scientific text. Journal of educational psychology, 81(2), 240.
33. Morch, N. J., Kjems, U., Hansen, L. K., Svarer, C., Law, I., Lautrup, B., and Rehm, K. (1995, November). Visualization of neural networks using saliency maps. In Proceedings of ICNN'95-International Conference on Neural Networks (Vol. 4, pp. 2085-2090). IEEE.
34. Nevatia, R., Hobbs, J., and Bolles, B. (2004, June). An ontology for video event representation. In 2004 Conference on Computer Vision and Pattern Recognition Workshop (pp. 119-119). IEEE.
35. A. Ramos-Soto and A. J. Bugarín and S. Barro and J. Taboada. Linguistic Descriptions for Automatic Generation of Textual Short-Term Weather Forecasts on Real Prediction Data IEEE Transactions on Fuzzy Systems 23(1), 44–57 (2015).
36. C. Rubio-Manzano, G. Trivino. Improving player experience in computer games by using players' behavior analysis and linguistic descriptions. Int. J. Hum.-Comput. Stud., 95 (2016), pp. 27?38.
37. E. Reiter, S. Sripada, J. Hunter, J. Yu, J and I. Davy. Choosing words in computer-generated weather forecasts Artificial Intelligence 167(1-2), 137–169 (2005).
38. Riedl, M. O., and Young, R. M. (2010). Narrative planning: Balancing plot and character. Journal of Artificial Intelligence Research, 39, 217-268.
39. Ribeiro, M. T., Singh, S., and Guestrin, C. (2016, August). " Why should i trust you?" Explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining (pp. 1135-1144).
40. Rieber, L. P. (1994). Computers graphics and learning. Brown & Benchmark Pub.
41. Skalski, P. (2019). Gentle Dive into Math Behind Convolutional Neural Networks. Online: https://towardsdatascience. com/gentle-dive-into-math-behind-convolutional-neural-networks-79a07dd44cf9, access: Jan, 30, 2021.
42. S. Sripada and E. Reiter and I. Davy. SumTime-Mousam: Configurable marine weather forecast generator. Expert Update 6(3), 4–10 (2003).
43. Samek, W., Binder, A., Montavon, G., Lapuschkin, S., and Müller, K. R. (2016). Evaluating the visualization of what a deep neural network has learned. IEEE transactions on neural networks and learning systems, 28(11), 2660-2673.
44. Shrikumar, A., Greenside, P., and Kundaje, A. (2017, July). Learning important features through propagating activation differences. In International Conference on Machine Learning (pp. 3145-3153). PMLR.
45. S. Sripada, N. Burnett, R. Turner, J. Mastin and D. Evans. A Case Study: NLG meeting Weather Industry Demand for Quality and Quantity of Textual Weather Forecasts Proceedings of the 8th International Natural Language Generation Conference (INLG), 1–5 (2014).
46. G. Trivino, M. Sugeno. Towards linguistic descriptions of phenomena. International Journal of Approximate Reasoning, 54(1), 22-34.
47. Trivino, G., and Sobrino, A. (2009). Human Perceptions versus Computational Perceptions in Computational Theory of Perceptions. In IFSA/EUSFLAT Conf. (pp. 327-332).
48. Tzeng, F. Y., and Ma, K. L. (2005). Opening the black box-data driven visualization of neural networks (pp. 383-390). IEEE.
49. Tversky, B., Morrison, J. B., and Betrancourt, M. (2002). Animation: can it facilitate?. International journal of human-computer studies, 57(4), 247-262.
50. Veale, M. (2017). Logics and practices of transparency and opacity in real-world applications of public sector machine learning. arXiv preprint arXiv:1706.09249.
51. VAN DER MEIJ, H., AND VAN DER MEIJ, J. A comparison of paper-based and video tutorials for software learning. *Computers and education 78* (2014), 150–159.

52. WINDERMERE, A. What is the importance of video tutorials to students?, 2019.
53. Zadeh, L. A. (1999). From computing with numbers to computing with words. From manipulation of measurements to manipulation of perceptions. IEEE Transactions on circuits and systems I: fundamental theory and applications, 46(1), 105-119.
54. Zadeh, L. A. (2008). Toward human level machine intelligence-is it achievable? the need for a paradigm shift. IEEE Computational Intelligence Magazine, 3(3), 11-22.
55. Zadeh, L. A. Computing With Words and Perceptions. A Paradigm Shift. In Joint Colloquium Distinguished Lecture Series June 22 (2009).