

# A Discussion about Explainable Inference on Sequential Data via Memory-Tracking

Biagio La Rosa<sup>1</sup>, Roberto Capobianco<sup>1,2</sup> and Daniele Nardi<sup>1</sup>

<sup>1</sup>*Sapienza University of Rome, Rome, Italy*

<sup>2</sup>*Sony AI*

## Abstract

The recent explosion of deep learning techniques boosted the application of Artificial Intelligence in a variety of domains, thanks to their high performance. However, performance comes at the cost of interpretability: deep models contain hundreds of nested non-linear operations that make it impossible to keep track of the chain of steps that bring to a given answer. In our recently published paper [1], we propose a method to improve the interpretability of a class of deep models, namely Memory Augmented Neural Networks (MANNs), when dealing with sequential data. Exploiting the capability of MANNs to store and access data in external memory, tracking the process, and connecting this information to the input sequence, our method extracts the most relevant sub-sequences that explain the answer. We evaluate our approach both on a modified T-maze [2, 3] and on the *Story Cloze Test* [4], obtaining promising results.

## Keywords

Explainable Artificial Intelligence, Deep Learning, Sequential Data.

## 1. Introduction

The availability of cheaper and more performing hardware, combined with the progress in the field of Machine Learning, made it possible to apply Artificial Intelligence techniques in a lot of domains, ranging from aerospace and justice to smartphones and domotics. Intelligent agents are now capable of learning complex tasks and adapting their behavior to new scenarios, often surpassing human performances [5]. Deep learning techniques are the cutting-edge technology behind these improvements: they are based on neural networks with hundreds of layers and millions of parameters, whose combination allows them to reach performances never seen before. Unfortunately, the high complexity of their structure makes also it impossible to inspect the exact chain of reasoning that brings the network to output a given answer. Hence, they are seen as black boxes, where one feeds an input and gets the output without the possibility to check the motivations behind their results.

The problem is exacerbated when dealing with sequential data, where recurrent or memory-augmented architectures have to be used. In these cases, the output depends not only on the

---

*AIXIA 2021 Discussion Papers*


✉ larosa@diag.uniroma1.it (B. La Rosa); capobianco@diag.uniroma1.it (R. Capobianco); nardi@diag.uniroma1.it (D. Nardi)

🌐 <https://biagiomattialarosa.github.io/> (B. La Rosa); <http://robertocapobianco.com/> (R. Capobianco)

🆔 0000-0002-4071-170X (B. La Rosa); 0000-0002-2219-215X (R. Capobianco); 0000-0001-6606-200X (D. Nardi)



© 2021 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

current observation but also on previous and future ones, making it hard to weigh the importance of each of them. The field of eXplainable Artificial Intelligence (XAI) tries to address these issues by proposing methods that are able to open the black box or build more transparent models. While there is abundant literature on methods for non-sequential data and classical models like Convolutional Neural Networks, the problem of interpretability of Memory Augmented Neural Networks when dealing with sequential data is less studied.

In our paper [1], we propose a method for getting explanations for a type of MANN, namely a simplified version of Differentiable Neural Computers (DNC) [6]. Our idea is to keep track of the memory usage, storing information about the accesses to the memory and its content, and connecting them to the input sequence, in order to retrieve the most relevant parts of the input associated with each prediction. The proposed method has several advantages, including the low computational cost and the high flexibility that allows to get different types of explanations in different contexts.

We evaluate our approach both on a modified T-maze [2, 3] and on the *Story Cloze Test* [4]. In the former, our results show that we are able to explain agent’s decisions. In the latter, we are able to reconstruct the most relevant premises that are used by the network to select the story endings. Additionally, we show that those premises are sufficient for the network to reproduce the inference – i.e., they are prime implicants [7].

The remainder of this paper is organized as follows. First, we review existing literature and discuss the state-of-the-art in network explainability (Section 2); then, we introduce our approach in Section 3 and present its experimental evaluation (Section 4). Finally, we discuss conclusions and future work.

## 2. Related Work

The proposed method is connected to the recent advances in Memory Augmented Neural Networks (MANNs). MANNs use an external memory to store and retrieve data during input processing. They can store past steps of a sequence, as in the case of recurrent architectures for sequential tasks, or they can store external knowledge in form of a knowledge base [8]. Thanks to their ability to store information for long periods, they often outperform LSTM networks, whose training is limited to data with smaller time windows. Usually, the network interacts with the memory through attention mechanisms, and it can also learn how to write and read the memory during the training process [9]. Differentiable Neural Computers [6] and End-To-End Memory Networks [10] are popular examples of this class of architectures. They are able to reach great results in multiple domains, such as visual question answering [11], image classification [12], reinforcement learning [13], and meta-learning [14]. Regarding the interpretability of MANNs, they are usually claimed as more interpretable by design, providing proof based on snapshot of the memory [6]. In this work we do a step forward by proposing a more structured way to inspect the mechanisms of these networks. According to Guidotti et al. [15] our work can be inserted in the literature about *outcome explanation methods* and attribution methods, which are a set of methods that try to give motivations behind specific predictions highlighting the contribution of input features towards the prediction. Examples of this category include saliency maps [16, 17, 18], local explainers [19, 20, 21] and ad-hoc

structures based on rules or weights [22, 23]. Our work can be placed in the latter category, since it is based on an ad-hoc module that exploits weights and inner mechanisms of the model to return explanations.

### 3. Method

In this section, we describe the architecture of Simplified Differentiable Neural Computers, the explanation module to compute the explanations, and the differences between our variant and the Differentiable Neural Computers (DNC) architecture [6].

#### 3.1. Simplified Differentiable Neural Computer

We propose a variant of DNC [6] called Simplified Differentiable Neural Computers. The architecture is composed of a controller  $f_c$  and an external memory  $\mathbf{M}$ . The memory is represented as a  $N \times C$  matrix, where  $N$  is the number of rows and  $C$  is the number of columns. During the training process, the controller learns how to write to and how to read information from external memory using attention mechanisms performed by  $R$  reading heads and  $W$  writing heads.

More specifically, in our case, the controller is an LSTM network, which takes as input a sequence of steps  $\mathbf{x} = [x_1, x_2, \dots, x_n]$  and returns a sequence of hidden states  $\mathbf{h} = [h_1, h_2, \dots, h_n]$ .

At each step  $t$ , the network chooses whether to add information into the memory  $\mathbf{M}_t$  and whether to read information from it, based on the value of the current hidden state  $h_t$ . Based on it, the network computes the following set of projections: the read keys  $\mathbf{k}_t^r$ , the write vectors  $\mathbf{v}_t^w$ , the write gates  $g_t^w$  (one for each head), the erase vectors  $\mathbf{e}_t^w$ , and the read strengths  $\beta_t^r$ , where the superscripts  $r$  and  $w$  are the index of the read head and the index of the write head, respectively.

The projections are then combined through attention mechanisms to perform writings and readings operations. The resulting attention weights associated to each operation are called *read weightings* and *write weightings*.

The read weightings are based on the cosine similarity between the controller output and the content of the memory, weighted by the read strength  $\beta_t^r$ :

$$\mathbf{w}_t^r = \text{cosine}(\mathbf{M}_t, \mathbf{k}_t^r, \beta_t^r) \quad (1)$$

The network uses the read weightings to perform readings, computing the read vectors  $\mathbf{r}_t^r$  as a weighted sum of the memory content:

$$\mathbf{r}_t^r = \mathbf{M}_t^T \mathbf{w}_t^r \quad (2)$$

The read vectors **actively** influence the inference process being interpolated with the input, and then fed to the last layer to compute the output  $\mathbf{y}_t$ .

$$\mathbf{y}_t = f_l([o_t, [r_t^1, \dots, r_t^R]]) \quad (3)$$

Differently, the writing process is mainly based on the previous usage of the memory. The usage is represented by a vector  $\mathbf{u}_t$  updated after each write, following the update rule:

$$\mathbf{u}_t = \mathbf{u}_{t-1} + \mathbf{w}_{t-1}^w \quad (4)$$

To compute the write weightings, the network multiplies the allocation weights  $\mathbf{a}_t$ , and the write gate  $g_t^w$ .

$$\mathbf{w}_i^w = g_t^w \mathbf{a}_t, \quad (5)$$

where the allocation weights are used to decide where to write the new information. Their computation is based on the usage vector  $\mathbf{u}_t$ :

$$\mathbf{a}_t[\phi_t[j]] = (1 - \mathbf{u}_t[\phi_t[j]]) \prod_{i=1}^{j-1} \mathbf{u}_t[\phi_t[i]] \quad (6)$$

where  $\phi_t$  is the sorted list of memory-cell indices in ascending order of usage. Intuitively, the write weightings are higher for cells never used before and low for cells recently written.

Finally, the write weightings are used to update the content of memory:

$$M_t = M_{t-1} \circ (1 - \mathbf{w}_i^w \mathbf{e}_t^T) + \mathbf{w}_i^w \mathbf{v}_t^w \quad (7)$$

where  $\mathbf{e}_t$  are the erase vectors that tell the system which cells can be freed,  $\mathbf{v}_t$  are the vectors to be written and  $\mathbf{w}_i$  are the write weightings.

### 3.2. Explanation module

The idea behind the explanation module is to exploit the writing and reading mechanisms to extract the information more often used by the network to compute the predictions. As mentioned before, the memory contains information extracted from the controller output, while the weightings describe the memory operations.

Let us start from the write weightings associated to a single cell: a high value indicates that the cell will contain most of the information of the current hidden state  $h_t$  (i.e. controller output) after the writing operation. Therefore, the explanation module creates a mapping between the cells with high weights and the current step  $t$ , storing the position of cells whose weightings are greater than the mean, and the input  $x_t$ . Note that tighter thresholds would not be more informative, since most of the time only one or few cells are written and many cells have write weightings close to zero. Since the network tries to write information on empty cells by design (Sect. 3.1), then the mapping between cells and inputs will be often one-to-one.

Regarding the read weightings, a high weight indicates that the cell contains information similar or related to the current state – due to the cosine similarity. It also means that the cell is heavily represented in the read vectors, and so it has a high impact on the inference process. In this case, the explanation module keeps track of the  $top-N_c$  read cells for each step, i.e., the  $N_c$  cells with the highest read weights.

At the end of the input parsing, the explanation module combines the information about written cells, the inputs, and the read cells to compute a history of the memory operations. Based on this history and on the characteristics of the task, it builds a rank of explanations by means of the frequency of input readings. Intuitively, since by design the readings are connected to the inference process, we expect that the inputs associated to the most read cells have a greater influence than the others, and so they can be considered as explanations, being the subsequences that steer the model during the inference process.

### 3.3. Differences with respect to DNCs

Before we proceed to show the results of our method, we want to highlight the key differences between the proposed variant and original DNC, and explain the motivations behind some decisions.

First of all, let us consider the writing operations. The DNCs use the cosine similarity to compute the write weightings in combination with allocation weights. In this way, during the parsing of the input, the network can overwrite or update information on a given cell. Conversely, the Simplified DNC uses only the allocation weights, avoiding overwriting cells in the memory as much as possible. This modification has not an impact on the performance as long as there is enough space to store information from the whole sequence, but it has a big impact on interpretability. Indeed, when states of several steps are stored in the same cell, then it becomes difficult to extract the single step responsible for the reading operation is being performed. Conversely, when the network is encouraged to write information on new cells, then each cell is likely to be only associated to one step, making the tracking and mapping of the inputs easier.

The second important difference is the lack of recurrence at the memory level. In DNCs, the input of LSTMs is the concatenation between the current step and the read vectors of the previous step, generating a recurrence at memory level and making the mapping between cells and steps harder. Indeed, in this case the output  $h_t$  of the LSTMs will be influenced not only from the previous steps due to the recurrence of LSTMs but also from the previous read steps, increasing the input dispersion and making the extraction of information harder. Moreover, in the case of memory recurrence, since we only use the cosine similarity to perform readings, the readings will tend to retrieve information from the same cells with little variation between two consecutive steps.

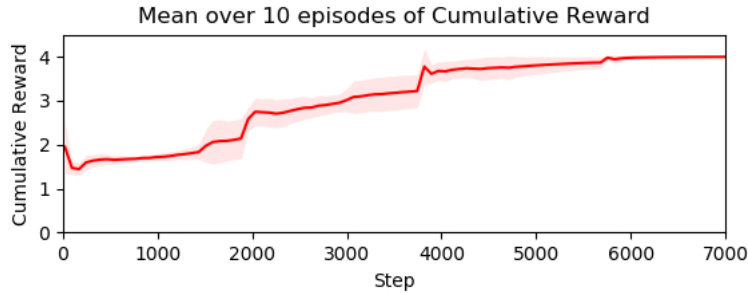
The last difference is the lack of temporal linkage addressing, whose inclusion makes the network heavier, while its effectiveness varies from problem to problem. This modification has no impact on the explanation module, since its working mechanisms are independent of the type of attention mechanism used to compute the weights.

## 4. Experiments

We divided this section into three subparts: the first describes the datasets and the protocols followed to evaluate the proposed method; the second presents the results both in terms of performance and explanations together with some examples showing what type of explanations can be obtained; and the last discusses the hyperparameters introduced by the method, its extension to LSTMs, and it presents a summary about its strengths and weaknesses.

### 4.1. Datasets

We test the proposed architecture on a modified version of the T-maze [2, 3] task and on the Story Cloze Test [4] dataset. T-maze is a non-Markovian discrete control task that evaluates an algorithm's ability to learn to remember observations and correlate events far apart in history. In our modified version, an agent has to move from the starting position to the T-junction through a corridor. At the T-junction, it must move either North or South to a changing goal



**Figure 1:** Mean and std of the cumulative reward obtained over 10 different training runs.

**Table 1**

Avg. accuracy comparison against task baseline. For [24], we report the scores from the original paper, while we compute our average accuracy over 10 different runs.

Avg. Accuracy		
Architecture	Dev	Test
[24]	77.12%	72.10%
DNC	71.30%	72.10%

position that depends on the symbol observed in a relevant step of the corridor. We add a symbol observable by the agent at every step of the corridor, but only one of them is relevant, and we specify the relevant corridor step in the starting position. If the agent takes the correct action at the junction, it receives a reward of 4, otherwise  $-0.1$ . In both cases, the episode terminates and a new episode is started. The Story Cloze Test is a commonsense reasoning framework for evaluating story understanding. In this task, an agent has to choose between two possible endings for a set of stories composed of four premises each.

## 4.2. Results

**Performance:** Fig. 1 and Table 1 show the performance of our architecture in both the tasks. Fig. 1 shows the average cumulative reward of the agent on the T-maze task over 10 different training runs. It shows that the model gradually converges to a solution in which the agent consistently achieves the highest reward. For the Story Cloze Test we follow the settings of Mihaylov et al. [24] and compare the performance against them in Table 1. Despite the fact that we use a smaller LSTM (128 units against 512) and we do not use some optimizations like the augmentation of the dataset, the system is able to reach the same generalization power, confirming the previous results of MANNs against LSTM models [6, 11, 12, 14].

**Explanations:** Now we will discuss how the method explained in Sect. 3.2 can be used to generate different types of explanations, tailored on the problem of interest and on our knowledge about the task.

In the case of Story Cloze Test, we are interested to extract the premises that bring the network

**Table 2**

Avg. explanation accuracy over 10 runs on the Story Cloze Test when feeding the model with only a random, the best and the worst premise.

Avg. Explanation Accuracy			
Datasets			
Premise	Train	Dev	Test
Random	57.5%	60.0%	55.6%
Best	66.0%	73.3%	63.8%
Worst	51.3%	53.3%	53.6%

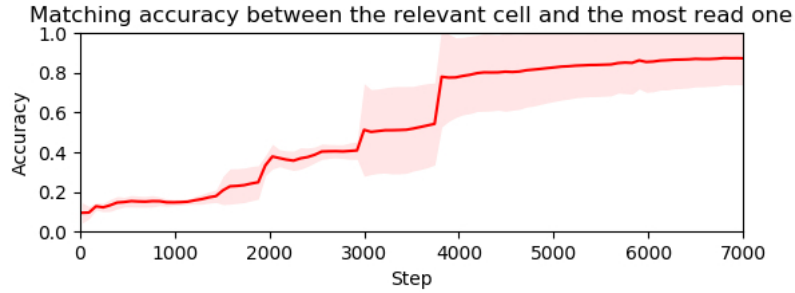
to choose the predicted ending of each story. During the parsing of the predicted ending, we expect that the network reads words of an important premise more often than those of an irrelevant one. Starting from this observation, the explanation module builds a rank of premises based on the frequency of readings of its words, attaching a score to each premise. From the rank, we can consider the top-ranked premise as the *best premise* for explaining the prediction, being the one that contains the most read words, and so it is often included in the read vectors that influenced the prediction. Contextually, we can identify the *worst premise* for explaining the prediction as the sub-sequence that contains the words associated with the minimum number of readings, and so ranked last.

To test the quality of the rank, we use the *explanation accuracy* metric defined as the times the model is able to reproduce its predictions by only using the provided premises. We expect that, if the ranking is meaningful, the explanation accuracy of the best premise will be greater than the one of the worst premise and the one of a random selected premise. In this case we can consider the best premise as an **explanation** of the predicted ending. Indeed, intuitively, if the explanation for a certain prediction is good, feeding that information alone to the network should result in the same prediction (and a good accuracy). Conversely, by only providing the worst explanation to the model, the prediction should change since the missing relevant information, lowering the explanation accuracy.

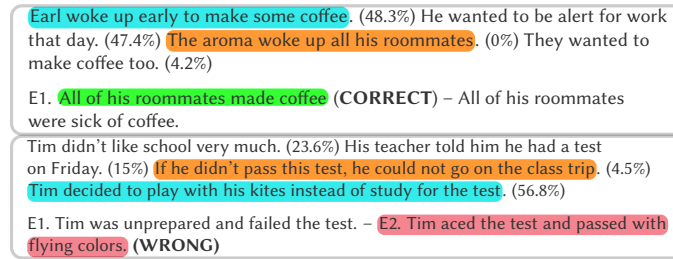
The results in Table 2 confirm the reliability of the ranking. Hence, the explanation accuracy of the best premise is always greater than the one of the worst premise, and the accuracy of the worst premise is lower than that of a randomly selected premise.

Figure 3 and Figure 4 show examples of the output of the explanation module. In particular, Figure 4 shows an example, extracted from the dataset, where the model returns the right prediction and the explanation module returns the premise rank. We can observe that the module focuses mainly on the third and fourth premises, assigning lower scores to the others. We can test the rank, modifying the input and observing how the model prediction and its confidence change. First of all, we can observe that modifying one of the two best premises the model becomes less confident about its prediction (a loss of about  $\sim 28\%$  and  $\sim 10\%$  respectively), while modifying the second premise it has no impact. Moreover, the importance of both of them is confirmed by the fact that we need to modify both in order to get a different prediction.

In the T-maze problem, we can exploit our prior knowledge about the problem. Here, the only



**Figure 2:** Matching accuracy between the relevant corridor step and the 2 most-read memory cells.



**Figure 3:** Example outputs on the Story Cloze Test. A relevance score is associated to each premise – ranked from best (blue) to worst (orange). Predictions are highlighted in green if correct, or red if wrong.

information that matters for predictions is the information stored in the relevant corridor step. Indeed, an agent can solve the task only learning to exploit it due to the nature of the task itself. Therefore, we use the explanation module to verify that the agent is acting correctly and is basing its decision on that information. To test this scenario, we check whether the information from the relevant corridor step is among the two most-read memory cells. We consider two cells instead of one to also account for the first corridor step, which could be used equally often. Figure 2 shows that this is the case. Moreover, comparing the trend against the learning curve shown in Figure 1, we can observe that the plots of cumulative reward and the matching accuracy follow the same behavior, meaning that, when the agent learns to exploit the relevant step corridor information stored in the memory, it improves its performance.

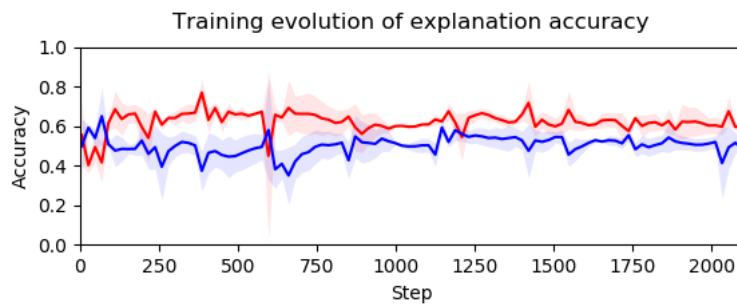
### 4.3. Additional insights

**Hyperparameters** The proposed module includes the  $N_c$  hyperparameter that controls how many cells consider for each reading. As shown in Table 3, it can have an impact on the goodness of returned explanation. For example, in the case of Story Cloze Test, using small  $N_c$  values is beneficial for the choice of the best premise. Indeed, in this way the module collects few cells for each step and, since these cells are those with the highest weights they have a direct influence on the prediction, it rewards the premises read in a more consistent way. Conversely, by setting  $N_c$  to a high value the rank will reward premises with the highest number of readings – independently of when the read occurred, introducing noise in the process. At the same time, when  $N_c$  is small, the information to choose the worst premise is very scarce while, when



P1. My best friend played a video game online.(4%)
P2. One day, she met a nice boy there.(21%)
P3. They talked every day and fell in love. (36%)
P4. They finally met in person and hit it off.(39%)
E1. They were very sad about the incident.(11.05)
E2. <b>The two became a very loving couple. (CORRECT)</b> (88.95)
Modified P4. They never met in person.
E1. They were very sad about the incident. (39.35.) ↑
E2. <b>The two became a very loving couple. (CORRECT)</b> (60.65) ↓
Modified P3. They never talked again.
E1. They were very sad about the incident. (22.32) ↑
E2. <b>The two became a very loving couple. (CORRECT)</b> (77.68) ↓
Modified P3. They never talked again.
Modified P4. They never met in person.
E1. <b>They were very sad about the incident. (WRONG)</b> (63.08) ↑
E2. The two became a very loving couple. (36.98) ↓
Modified P2. One day, she played another game.
E1. They were very sad about the incident. (11.05) =
E2. <b>The two became a very loving couple. (CORRECT)</b> (88.95) =

**Figure 4:** A sample extracted from the dataset. A relevance score is associated to each premise (first square). The other squares show how the prediction of the model changes when you modify one or more premises. Predictions are highlighted in green if correct, or red if wrong.



**Figure 5:** Explanation accuracy over training. Metrics are computed when using only the best (red line) and the worst (blue line) premise as input.

$N_c$  is large, it is more reliable. For this reason, this hyperparameter should be tuned in the validation process or fixed using prior knowledge, and it represents a trade-off to be taken into consideration. Additionally, the plot in Figure 5 indicates that explanations provided by our system are meaningful from the very first training steps and, consequently, that the network quickly learns to use its memory.

**Extension to LSTMs** A natural question is whether we can apply the same approach to LSTM networks, thus exploiting their internal memory. Unfortunately, the answer is negative due to the way in which they use the memory. Indeed, during the parsing of the input, the hidden state  $h_t$  is only used once as additional input for the next time step, eventually merged inside the cell state, and then discarded. Because only the cell states store information, and they include

**Table 3**

Avg. explanation accuracy on the dev set for the best and worst premises using different  $N_c$  values. We highlight in bold the best and worst performance respectively, for each premise type among the different threshold values.

Avg. Explanation Accuracy					
Premise	Threshold				
	Top1	Top5	Top10	Top25	>Median
Best	<b>75.6%</b>	<b>75.6%</b>	73.3%	71.1%	40.0%
Worst	60.0%	60.0%	53.3%	<b>40.0%</b>	48.9%

aggregated information from several steps in the sequence, it becomes impossible to reconstruct the relevant information. A possible alternative is to directly apply the cosine similarity over the resulting states, selecting the top  $x$  states similar to the state of the prediction step, and then compute the explanations. The drawback in this case is that one has to decide a priori how many words to consider in the explanation, thus introducing a hyperparameter difficult to tune. Moreover, the most similar states are not actively used in the inference process by the LSTM, therefore the connection between them and the predictions would be unclear and undefined.

**Limitations and strengths** Summing up, the strengths of this approach are several. It has a low computational cost, because it needs only space for storing the memory history and a negligible time to compute the rank. Its incremental nature makes it ready for online scenarios, where the full sequence is often lacking. It is flexible, allowing to shape explanations on the basis of the problem and the query of interest. Nevertheless, there are some drawbacks connected to MANNs and the  $N_c$  hyperparameter to consider. The training process of MANNs is often unstable, showing a large variance and an high sensitive to hyper-parameter choices, making them hard to tune and apply to new domains [25]. We can observe a similar behavior on the  $N_c$  hyperparameter and, more in general, on how the explanation module selects the cells to monitor. Currently, the choice must be done using a priori knowledge about the problem or validating it in a development set, due to the lack of a general rule. This means that one can obtain different results and explanations by simply modifying the way in which the module keeps track of the memory. Moreover, the network potentially could learn to ignore the memory content on some predictions and it is hard to understand when this happens. To limit the extent of this problem we apply the Bypass Dropout [25] to the link between controller and output, but it remains an open problem to be solved in future research.

## 5. Conclusion and future work

In this paper, we discussed a method presented in our previous work [1] to get explanations on a neural network augmented with external memory. The problem of interpretability is becoming crucial for Artificial Intelligence and improvements in this area can shape the future

of society itself. The weaknesses and strengths of this approach allowed us to further develop some of the concepts behind the paper, finding applications in the image classification domain, improving both the performance and the interpretability [26]. However, we think that the combination between memory, neural networks, and explainability has not been fully exploited and understood yet, leaving space for future research.

## References

- [1] B. La Rosa, R. Capobianco, D. Nardi, Explainable inference on sequential data via memory-tracking, in: Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, International Joint Conferences on Artificial Intelligence Organization, 2020. doi:10.24963/ijcai.2020/278.
- [2] B. Bakker, Reinforcement learning with long short-term memory, in: Advances in Neural Information Processing Systems, 2002, pp. 1475–1482.
- [3] D. Wierstra, A. Foerster, J. Peters, J. Schmidhuber, Solving deep memory pomdps with recurrent policy gradients, in: Int. Conf. on Artificial Neural Networks, Springer, 2007, pp. 697–706.
- [4] N. Mostafazadeh, N. Chambers, X. He, D. Parikh, D. Batra, L. Vanderwende, P. Kohli, J. Allen, A corpus and cloze evaluation for deeper understanding of commonsense stories, in: Proc. of the 2016 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego, California, 2016, pp. 839–849. URL: <https://www.aclweb.org/anthology/N16-1098>. doi:10.18653/v1/N16-1098.
- [5] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. Lillicrap, D. Silver, Mastering atari, go, chess and shogi by planning with a learned model, Nature 588 (2020) 604–609. doi:10.1038/s41586-020-03051-4.
- [6] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou, et al., Hybrid computing using a neural network with dynamic external memory, Nature 538 (2016) 471.
- [7] A. Shih, A. Choi, A. Darwiche, A symbolic approach to explaining bayesian network classifiers, in: Proc. of the Twenty-Seventh International Joint Conf. on Artificial Intelligence, 2018. doi:10.24963/ijcai.2018/708.
- [8] E. Dinan, S. Roller, K. Shuster, A. Fan, M. Auli, J. Weston, Wizard of wikipedia: Knowledge-powered conversational agents (2018). arXiv:1811.01241.
- [9] A. Graves, G. Wayne, I. Danihelka, Neural turing machines, arXiv preprint arXiv:1410.5401 (2014).
- [10] S. Sukhbaatar, a. szlam, J. Weston, R. Fergus, End-to-end memory networks, in: C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, R. Garnett (Eds.), Advances in Neural Information Processing Systems, volume 28, Curran Associates, Inc., 2015. URL: <https://proceedings.neurips.cc/paper/2015/file/8fb21ee7a2207526da55a679f0332de2-Paper.pdf>.
- [11] C. Ma, C. Shen, A. Dick, Q. Wu, P. Wang, A. van den Hengel, I. Reid, Visual question answering with memory-augmented networks, in: 2018 IEEE/CVF Conf. on Computer Vision and Pattern Recognition, 2018. doi:10.1109/cvpr.2018.00729.

- [12] Q. Cai, Y. Pan, T. Yao, C. Yan, T. Mei, Memory matching networks for one-shot image recognition, in: 2018 IEEE/CVF Conf. on Computer Vision and Pattern Recognition, 2018. doi:10.1109/cvpr.2018.00429.
- [13] A. Pritzel, B. Uria, S. Srinivasan, A. Puigdomènech, O. Vinyals, D. Hassabis, D. Wierstra, C. Blundell, Neural episodic control (2017). arXiv:1703.01988.
- [14] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, T. P. Lillicrap, One-shot learning with memory-augmented neural networks, CoRR abs/1605.06065 (2016). URL: <http://arxiv.org/abs/1605.06065>.
- [15] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, D. Pedreschi, A survey of methods for explaining black box models, ACM Computing Surveys 51 (2018) 1–42. doi:10.1145/3236009.
- [16] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, A. Torralba, Learning deep features for discriminative localization, in: 2016 IEEE Conf. on Computer Vision and Pattern Recognition, 2016. doi:10.1109/cvpr.2016.319.
- [17] D. Baehrens, T. Schroeter, S. Harmeling, M. Kawanabe, K. Hansen, K.-R. Müller, How to explain individual classification decisions, J. Mach. Learn. Res. 11 (2010) 1803–1831. URL: <http://dl.acm.org/citation.cfm?id=1756006.1859912>.
- [18] M. Sundararajan, A. Taly, Q. Yan, Axiomatic attribution for deep networks (2017). arXiv:1703.01365.
- [19] M. Ribeiro, S. Singh, C. Guestrin, “why should i trust you?”: Explaining the predictions of any classifier, in: Proc. of the 2016 Conf. of the North American Chapter of the Association for Computational Linguistics: Demonstrations, 2016. doi:10.18653/v1/n16-3020.
- [20] S. M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, in: Advances in Neural Information Processing Systems, 2017, pp. 4765–4774.
- [21] M. T. Ribeiro, S. Singh, C. Guestrin, Anchors: High-precision model-agnostic explanations, in: AAAI Conf. on Artificial Intelligence, 2018.
- [22] J. T. Springenberg, A. Dosovitskiy, T. Brox, M. A. Riedmiller, Striving for simplicity: The all convolutional net, CoRR abs/1412.6806 (2014).
- [23] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, W. Samek, On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation, PLOS ONE 10 (2015) e0130140. doi:10.1371/journal.pone.0130140.
- [24] T. Mihaylov, A. Frank, Story cloze ending selection baselines and data examination, LSDSem 2017 (2017) 87.
- [25] J. Franke, J. Niehues, A. Waibel, Robust and scalable differentiable neural computer for question answering, in: Proc. of the Workshop on Machine Reading for Question Answering, 2018, pp. 47–59.
- [26] B. La Rosa, R. Capobianco, D. Nardi, Memory wrap: a data-efficient and interpretable extension to image classification models (2021). arXiv:2106.01440.