Working Memory Networks: Augmenting Memory Networks with a Relational Reasoning Module

Juan Pavez*, Héctor Allende
Department of Informatics
Federico Santa María
Technical University
Valparaíso, Chile
juan.pavezs@alumnos.usm.cl
hallende@inf.utfsm.cl

Héctor Allende-Cid
Escuela de Ingeniería Informática
Pontífica Universidad Católica
de Valparaíso
Valparaíso, Chile
hector.allende@pucv.cl

Abstract

During the last years, there has been a lot of interest in achieving some kind of complex reasoning using deep neural networks. To do that, models like Memory Networks (MemNNs) have combined external memory storages and attention mechanisms. These architectures, however, lack of more complex reasoning mechanisms that could allow, for instance, relational reasoning. Relation Networks (RNs), on the other hand, have shown outstanding results in relational reasoning tasks. Unfortunately, their computational cost grows quadratically with the number of memories, something prohibitive for larger problems. To solve these issues, we introduce the Working Memory Network, a MemNN architecture with a novel working memory storage and reasoning module. Our model retains the relational reasoning abilities of the RN while reducing its computational complexity from quadratic to linear. We tested our model on the text QA dataset bAbI and the visual QA dataset NLVR. In the jointly trained bAbI-10k, we set a new state-of-the-art, achieving a mean error of less than 0.5%. Moreover, a simple ensemble of two of our models solves all 20 tasks in the joint version of the benchmark.

1 Introduction

A central ability needed to solve daily tasks is complex reasoning. It involves the capacity to comprehend and represent the environment, retain information from past experiences, and solve problems based on the stored information. Our ability to solve those problems is supported by multiple specialized components, including short-term memory storage, long-term semantic and procedural memory, and an executive controller that, among others, controls the attention over memories (Baddeley, 1992).

Many promising advances for achieving complex reasoning with neural networks have been obtained during the last years. Unlike symbolic approaches to complex reasoning, deep neural networks can learn representations from perceptual information. Because of that, they do not suffer from the symbol grounding problem (Harnad, 1999), and can generalize better than classical symbolic approaches. Most of these neural network models make use of an explicit memory storage and an attention mechanism. For instance, Memory Networks (MemNN), Dynamic Memory Networks (DMN) or Neural Turing Machines (NTM) (Weston et al., 2014; Kumar et al., 2016; Graves et al., 2014) build explicit memories from the perceptual inputs and access these memories using learned attention mechanisms. After that some memories have been attended, using a multi-step procedure, the attended memories are combined and passed through a simple output layer that produces a final answer. While this allows some multi-step inferential process, these networks lack a more complex reasoning mechanism, needed for more elaborated tasks such as inferring relations among entities (relational reasoning). On the contrary, Relation Networks (RNs), proposed in Santoro et al. (2017), have shown outstanding performance in relational reasoning tasks. Nonetheless, a major drawback of RNs is that they consider each of the input objects in pairs, having to process a quadratic number of relations. That limits the usability of the model on large problems and makes forward and backward computations quite expensive. To solve these problems we propose a novel Memory Network

Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Long Papers), pages 1000–1009
Melbourne, Australia, July 15 - 20, 2018.

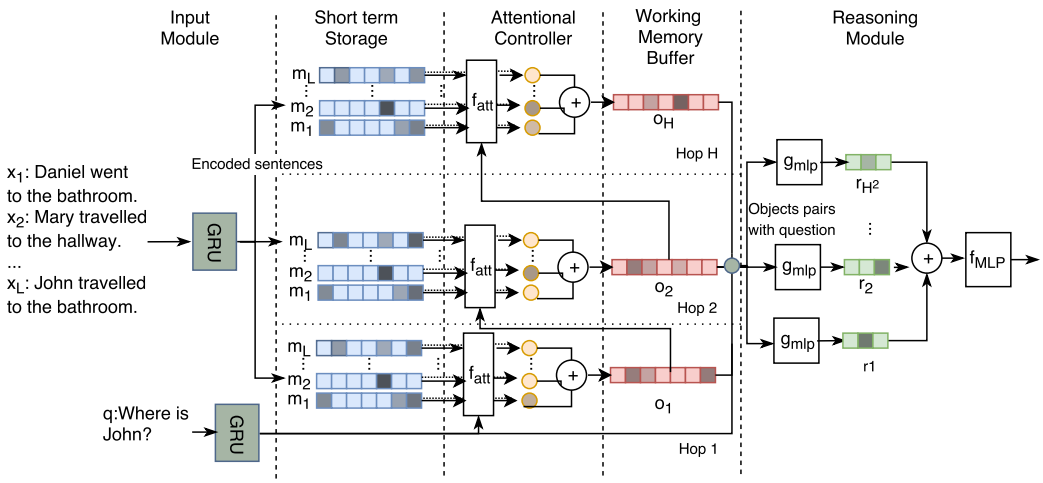


Figure 1: The W-MemNN model applied to textual question answering. Each input fact is processed using a GRU, and the output representation is stored in the short-term memory storage. Then, the attentional controller computes an output vector that summarizes relevant parts of the memories. This process is repeated H hops (a dotted line delimits each hop), and each output is stored in the working memory buffer. Finally, the output of each hop is passed to the reasoning module that produces the final output.

architecture called the Working Memory Network (W-MemNN). Our model augments the original MemNN with a relational reasoning module and a new working memory buffer.

The attention mechanism of the Memory Network allows the filtering of irrelevant inputs, reducing a lot of the computational complexity while keeping the relational reasoning capabilities of the RN. Three main components compose the W-MemNN: An input module that converts the perceptual inputs into an internal vector representation and save these representations into a short-term storage, an attentional controller that attend to these internal representations and update a working memory buffer, and a reasoning module that operates on the set of objects stored in the working memory buffer in order to produce a final answer. This component-based architecture is inspired by the well-known model from cognitive sciences called the multi-component working memory model, proposed in Baddeley and Hitch (1974).

We studied the proposed model on the text-based QA benchmark bAbI (Weston et al., 2015) which consists of 20 different toy tasks that measure different reasoning skills. While models such as EntNet (Henaff et al., 2016) have focused on the per-task training version of the benchmark (where a different model is trained for each task), we decided to focus on the jointly trained version of the task, where the model is trained on all tasks simultaneously. In the jointly trained bAbI-10k benchmark we achieved state-of-the-art performance, improving the previous state-of-the-art on more than 2%. Moreover, a simple ensemble of two of our models can solve all 20 tasks simultaneously. Also, we tested our model on the visual QA dataset NLVR. In that dataset, we obtained performance at the level of the Module Neural Networks (Andreas et al., 2016). Our model, however, achieves these results using the raw input statements, without the extra text processing used in the Module Networks.

Finally, qualitative and quantitative analysis shows that the inclusion of the Relational Reasoning module is crucial to improving the performance of the MemNN on tasks that involve relational reasoning. We can achieve this performance by also reducing the computation times of the RN considerably. Consequently, we hope that this contribution may allow applying RNs to larger problems.

2 Model

Our model is based on the Memory Network architecture. Unlike MemNN we have included a reasoning module that helps the network to solve more complex tasks. The proposed model consists of three main modules: An input module, an at-

tentional controller, and a reasoning module. The model processes the input information in multiple passes or hops. At each pass the output of the previous hop can condition the current pass, allowing some incremental refinement.

Input module: The input module converts the perceptual information into an internal feature representation. The input information can be processed in chunks, and each chunk is saved into a short-term storage. The definition of what is a chunk of information depends on each task. For instance, for textual question answering, we define each chunk as a sentence. Other options might be n-grams or full documents. This short-term storage can only be accessed during the hop.

Attentional Controller: The attentional controller decides in which parts of the short-term storage the model should focus. The attended memories are kept during all the hops in a working memory buffer. The attentional controller is conditioned by the task at hand, for instance, in question answering the question can condition the attention. Also, it may be conditioned by the output of previous hops, allowing the model to change its focus to new portions of the memory over time. Many models compute the attention for each memory using a compatibility function between the memory and the question. Then, the output is calculated as the weighted sum of the memory values, using the attention as weight. A simple way to compute the attention for each memory is to use dot-product attention. This kind of mechanism is used in the original Memory Network and computes the attention value as the dot product between each memory and the question. Although this kind of attention is simple, it may not be enough for more complex tasks. Also, since there are no learned weights in the attention mechanism, the attention relies entirely on the learned embeddings. That is something that we want to avoid in order to separate the learning of the input and attention module. One way to allow learning in the dot-product attention is to project the memories and query vectors linearly. That is done by multiplying each vector by a learned projection matrix (or equivalently a feed-forward neural network). In this way, we can set apart the attention and input embeddings learning, and also allow more complex patterns of attention.

Reasoning Module: The memories stored in the working memory buffer are passed to the reasoning module. The choice of reasoning mechanism is left open and may depend on the task at hand. In this work, we use a Relation Network as the reasoning module. The RN takes the attended memories in pairs to infer relations among the memories. That can be useful, for example, in tasks that include comparisons.

A detailed description of the full model is shown in Figure 1.

2.1 W-MemN2N for Textual Question Answering

We proceed to describe an implementation of the model for textual question answering. In textual question answering the input consists of a set of sentences or facts, a question, and an answer. The goal is to answer the question correctly based on the given facts.

Let (s, q, a) represents an input sample, consisting of a set of sentences $s = \{x_i\}_{i=1}^{L}$, a query q and an answer a. Each sentence contains M words, $\{w_i\}_{i=1}^{M}$, where each word is represented as a one-hot vector of length $|V|$, being $|V|$ the vocabulary size. The question contains Q words, represented as in the input sentences.

Input Module

Each word in each sentence is encoded into a vector representation v_i using an embedding matrix $W \in \mathbb{R}^{|V| \times d}$, where d is the embedding size. Then, the sentence is converted into a memory vector m_i using the final output of a gated recurrent neural network (GRU) (Chung et al., 2014):

$$m_i = \text{GRU}([v_1, v_2, ..., v_M])$$

Each memory $\{m_i\}_{i=1}^{L}$, where $m_i \in \mathbb{R}^d$, is stored into the short-term memory storage. The question is encoded into a vector u in a similar way, using the output of a gated recurrent network.

Attentional Controller

Our attention module is based on the Multi-Head attention mechanism proposed in Vaswani et al. (2017). First, the memories are projected using a projection matrix $W_m \in \mathbb{R}^{d \times d}$, as $m'_i = W_m m_i$. Then, the similarity between the projected memory and the question is computed using the Scaled Dot-Product attention:

$$\alpha_i = \text{Softmax}\left(\frac{u^T m'_i}{\sqrt{d}}\right) \quad (1)$$

$$= \frac{\exp((u^T m'_i)/\sqrt{d})}{\sum_j \exp((u^T m'_j)/\sqrt{d})}. \quad (2)$$

Next, the memories are combined using the attention weights α_i, obtaining an output vector $h = \sum_j \alpha_j m_j$.

In the Multi-Head mechanism, the memories are projected S times using different projection matrices $\{W_m^s\}_{s=1}^S$. For each group of projected memories, an output vector $\{h_i\}_{i=1}^S$ is obtained using the Scaled Dot-Product attention (eq. 2). Finally, all vector outputs are concatenated and projected again using a different matrix:

$$o_k = [h_1; h_2; ...; h_S]W_o,$$

where ; is the concatenation operator and $W_o \in \mathbb{R}^{Sd \times d}$. The o_k vector is the final response vector for the hop k. This vector is stored in the working memory buffer. The attention procedure can be repeated many times (or hops). At each hop, the attention can be conditioned on the previous hop by replacing the question vector u by the output of the previous hop. To do that we pass the output through a simple neural network f_t. Then, we use the output of the network as the new conditioner:

$$o_k^n = f_t(o_k). \tag{3}$$

This network allows some learning in the transition patterns between hops.

We found Multi-Head attention to be very useful in the joint bAbI task. This can be a product of the intrinsic multi-task nature of the bAbI dataset. A possibility is that each attention head is being adapted for different groups of related tasks. However, we did not investigate this further.

Also, note that while in this section we use the same set of memories at each hop, this is not necessary. For larger sequences each hop can operate in different parts of the input sequence, allowing the processing of the input in various steps.

Reasoning Module

The outputs stored in the working memory buffer are passed to the reasoning module. The reasoning module used in this work is a Relation Network (RN). In the RN the output vectors are concatenated in pairs together with the question vector. Each pair is passed through a neural network g_θ and all the outputs of the network are added to produce a single vector. Then, the sum is passed to a final neural network f_ϕ:

$$r = f_\phi\left(\sum_{i,j} g_\theta([o_i; o_j; u])\right), \tag{4}$$

The output of the Relation Network is then passed through a final weight matrix and a softmax to produce the predicted answer:

$$\hat{a} = \text{Softmax}(Vr), \tag{5}$$

where $V \in \mathbb{R}^{|A| \times d_\phi}$, $|A|$ is the number of possible answers and d_ϕ is the dimension of the output of f_ϕ. The full network is trained end-to-end using standard cross-entropy between $\hat{a}$ and the true label a.

3 Related Work

3.1 Memory Augmented Neural Networks

During the last years, there has been plenty of work on achieving complex reasoning with deep neural networks. An important part of these developments has used some kind of explicit memory and attention mechanisms. One of the earliest recent work is that of Memory Networks (Weston et al., 2014). Memory Networks work by building an addressable memory from the inputs and then accessing those memories in a series of reading operations. Another, similar, line of work is the one of Neural Turing Machines. They were proposed in Graves et al. (2014) and are the basis for recent neural architectures including the Differentiable Neural Computer (DNC) and the Sparse Access Memory (SAM) (Graves et al., 2016; Rae et al., 2016). The NTM model also uses a content addressable memory, as in the Memory Network, but adds a write operation that allows updating the memory over time. The management of the memory, however, is different from the one of the MemNN. While the MemNN model pre-load the memories using all the inputs, the NTM writes and read the memory one input at a time.

An additional model that makes use of explicit external memory is the Dynamic Memory Network (DMN) (Kumar et al., 2016; Xiong et al., 2016). The model shares some similarities with the Memory Network model. However, unlike the MemNN model, it operates in the input sequentially (as in the NTM model). The model defines an Episodic Memory module that makes use of a Gated Recurrent Neural Network (GRU) to store and update an internal state that represents the episodic storage.

3.2 Memory Networks

Since our model is based on the MemNN architecture, we proceed to describe it in more detail. The

Memory Network model was introduced in Weston et al. (2014). In that work, the authors proposed a model composed of four components: The input feature map that converts the input into an internal vector representation, the generalization module that updates the memories given the input, the output feature map that produces a new output using the stored memories, and the response module that produces the final answer. The model, as initially proposed, needed some strong supervision that explicitly tells the model which memories to attend. In order to solve that limitation, the End-To-End Memory Network (MemN2N) was proposed in Sukhbaatar et al. (2015).

The model replaced the hard-attention mechanism used in the original MemNN by a soft-attention mechanism that allowed to train it end-to-end without strong supervision. In our model, we use a component-based approach, as in the original MemNN architecture. However, there are some differences: First, our model makes use of two external storages: a short-term storage, and a working memory buffer. The first is equivalent to the one updated by the input and generalization module of the MemNN. The working memory buffer, on the other hand, does not have a counterpart in the original model. Second, our model replaces the response module by a reasoning module. Unlike the original MemNN, our reasoning module is intended to make more complex work than the response module, that was only designed to produce a final answer.

3.3 Relation Networks

The ability to infer and learn relations between entities is fundamental to solve many complex reasoning problems. Recently, a number of neural network models have been proposed for this task. These include Interaction Networks, Graph Neural Networks, and Relation Networks (Battaglia et al., 2016; Scarselli et al., 2009; Santoro et al., 2017). In specific, Relation Networks (RNs) have shown excellent results in solving textual and visual question answering tasks requiring relational reasoning. The model is relatively simple: First, all the inputs are grouped in pairs and each pair is passed through a neural network. Then, the outputs of the first network are added, and another neural network processes the final vector. The role of the first network is to infer relations among each pair of objects. In Palm et al. (2017) the authors propose a recurrent extension to the RN. By allowing multiple steps of relational reasoning, the model can learn to solve more complex tasks. The main issue with the RN architecture is that its scale very poorly for larger problems. That is because it operates on $O(n^2)$ pairs, where n is the number of input objects (for instance, sentences in the case of textual question answering). This becomes quickly prohibitive for tasks involving many input objects.

3.4 Cognitive Science

The concept of working memory has been extensively developed in cognitive psychology. It consists of a limited capacity system that allows temporary storage and manipulation of information and is crucial to any reasoning task. One of the most influential models of working memory is the multi-component model of working memory proposed by Baddeley and Hitch (1974). This model is composed both of a supervisory attentional controller (the central executive) and two short-term storage systems: The phonological loop, capable of holding speech-based information, and the visuospatial sketchpad, concerned with visual storage. The central executive plays various functions, including the capacity to focus attention, to divide attention and to control access to long-term memory. Later modifications to the model (Baddeley, 2000) include an episodic buffer that is capable of integrating and holding information from different sources. Connections of the working memory model to memory augmented neural networks have been already studied in Graves et al. (2014). We follow this effort and subdivide our model into components that resemble (in a basic way) the multi-component model of working memory. Note, however, that we use the term working memory buffer instead of episodic buffer. That is because the episodic buffer has an integration function that our model does not cover. However, that can be an interesting source of inspiration for next versions of the model that integrate both visual and textual information for question answering.

4 Experiments

4.1 Textual Question Answering

To evaluate our model on textual question answering we used the Facebook bAbI-10k dataset (Weston et al., 2015). The bAbI dataset is a textual

	LSTM	MN-S	MN	SDNC	WMN	WMN†
1: 1 supporting fact	0.0	0.0	0.0	0.0	0.0	0.0
2: 2 supporting facts	81.9	0.0	1.0	0.6	0.7	0.3
3: 3 supporting facts	83.1	0.0	6.8	0.7	5.3	4.6
4: 2 argument relations	0.2	0.0	0.0	0.0	0.0	0.0
5: 3 argument relations	1.2	0.3	6.1	0.3	0.6	0.4
6: yes/no questions	51.8	0.0	0.1	0.0	0.0	0.0
7: counting	24.9	3.3	6.6	0.2	0.6	0.5
8: lists/sets	34.1	1.0	2.7	0.2	0.2	0.3
9: simple negation	20.2	0.0	0.0	0.0	0.0	0.0
10: indefinite knowledge	30.1	0.0	0.5	0.2	0.5	0.0
11: basic coreference	10.3	0.0	0.0	0.0	0.3	0.0
12: conjunction	23.4	0.0	0.1	0.1	0.0	0.0
13: compound coreference	6.1	0.0	0.0	0.1	0.0	0.0
14: time reasoning	81.0	0.0	0.0	0.1	0.0	0.0
15: basic deduction	78.7	0.0	0.2	0.0	0.0	0.0
16: basic induction	51.9	0.0	0.2	54.1	0.0	0.3
17: positional reasoning	50.1	24.6	41.8	0.3	0.3	0.1
18: size reasoning	6.8	2.1	8.0	0.1	0.1	0.4
19: path finding	90.3	31.9	75.7	1.2	0.6	0.0
20: agent's motivations	2.1	0.	0.0	0.0	0.0	0.0
Mean Error (%)	36.4	3.2	7.5	2.8	0.4	**0.3**
Failed tasks (err. $> 5\%$)	16	2	6	1	1	**0**

Table 1: Test accuracies on the jointly trained bAbI-10k dataset. `MN-S` stands for strongly supervised Memory Network, `MN-U` for end-to-end Memory Network without supervision, and `WMN` for Working Memory Network. Results for `LSTM`, `MN-U`, and `MN-S` are took from Sukhbaatar et al. (2015). Results for `SDNC` are took from Rae et al. (2016). `WMN`† is an ensemble of two Working Memory Networks.

QA benchmark composed of 20 different tasks. Each task is designed to test a different reasoning skill, such as deduction, induction, and coreference resolution. Some of the tasks need relational reasoning, for instance, to compare the size of different entities. Each sample is composed of a question, an answer, and a set of facts. There are two versions of the dataset, referring to different dataset sizes: bAbI-1k and bAbI-10k. In this work, we focus on the bAbI-10k version of the dataset which consists of $10,000$ training samples per task. A task is considered solved if a model achieves greater than 95% accuracy. Note that training can be done per-task or joint (by training the model on all tasks at the same time). Some models (Liu and Perez, 2017) have focused in the per-task training performance, including the EntNet model (Henaff et al., 2016) that solves all the tasks in the per-task training version. We choose to focus on the joint training version since we think is more indicative of the generalization properties of the model. A detailed analysis of the dataset can be found in Lee et al. (2015).

Model Details

To encode the input facts we used a word embedding that projected each word in a sentence into a real vector of size d. We defined $d = 30$ and used a GRU with 30 units to process each sentence. We used the 30 sentences in the support set that were immediately prior to the question. The question was processed using the same configuration but with a different GRU. We used 8 heads in the Multi-Head attention mechanism. For the transition networks f_t, which operates in the output of each hop, we used a two-layer MLP consisting of 15 and 30 hidden units (so the output preserves the memory dimension). We used $H = 4$ hops (or equivalently, a working memory buffer of size 4). In the reasoning module, we used a 3-layer MLP consisting of 128 units in each layer and with ReLU non-linearities for g_θ. We omitted the f_ϕ network since we did not observe improvements when using it. The final layer was a linear layer that produced logits for a softmax over the

answer vocabulary.

Training Details

We trained our model end-to-end with a cross-entropy loss function and using the Adam optimizer (Kingma and Ba, 2014). We used a learning rate of $\nu = 1e^{-3}$. We trained the model during 400 epochs. For training, we used a batch size of 32. As in Sukhbaatar et al. (2015) we did not average the loss over a batch. Also, we clipped gradients with norm larger than 40 (Pascanu et al., 2013). For all the dense layers we used ℓ_2 regularization with value $1e^{-3}$. All weights were initialized using Glorot normal initialization (Glorot and Bengio, 2010). 10% of the training set was held-out to form a validation set that we used to select the architecture and for hyperparameter tunning. In some cases, we found useful to restart training after the 400 epochs with a smaller learning rate of $1e^{-5}$ and anneals every 5 epochs by $\nu/2$ until 20 epochs were reached.

bAbI-10k Results

On the jointly trained bAbI-10k dataset our best model (out of 10 runs) achieves an accuracy of **99.58%**. That is a **2.38%** improvement over the previous state-of-the-art that was obtained by the Sparse Differential Neural Computer (SDNC) (Rae et al., 2016). The best model of the 10 runs solves almost all tasks of the bAbI-10k dataset (by a 0.3% margin). However, a simple ensemble of the best two models solves all 20 tasks and achieves an almost perfect accuracy of **99.7%**. We list the results for each task in Table 1. Other authors have reported high variance in the results, for instance, the authors of the SDNC report a mean accuracy and standard deviation over 15 runs of 93.6 ± 2.5 (with 15.9 ± 1.6 passed tasks). In contrast, our model achieves a mean accuracy of 98.3 ± 1.2 (with 18.6 ± 0.4 passed tasks), which is better and more stable than the average results obtained by the SDNC.

The Relation Network solves 18/20 tasks. We achieve even better performance, and with considerably fewer computations, as is explained in Section 4.3. We think that by including the attention mechanism, the relation reasoning module can focus on learning the relation among relevant objects, instead of learning spurious relations among irrelevant objects. For that, the Multi-Head attention mechanism was very helpful.

The Effect of the Relational Reasoning Module

When compared to the original Memory Network, our model substantially improves the accuracy of tasks 17 (positional reasoning) and 19 (path finding). Both tasks require the analysis of multiple relations (Lee et al., 2015). For instance, the task 19 needs that the model reasons about the relation of different positions of the entities, and in that way find a path to arrive from one to another. The accuracy improves in **75.1%** for task 19 and in **41.5%** for task 17 when compared with the MemN2N model. Since both tasks require reasoning about relations, we hypothesize that the relational reasoning module of the W-MemNN was of great help to improve the performance on both tasks.

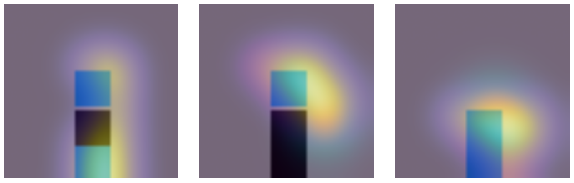
The Relation Network, on the other hand, fails in the tasks 2 (2 supporting facts) and 3 (3 supporting facts). Both tasks require handling a significant number of facts, especially in task 3. In those cases, the attention mechanism is crucial to filter out irrelevant facts.

4.2 Visual Question Answering

To further study our model we evaluated its performance on a visual question answering dataset. For that, we used the recently proposed NLVR dataset (Suhr et al., 2017). Each sample in the NLVR dataset is composed of an image with three sub-images and a statement. The task consists in judging if the statement is true or false for that image. Evaluating the statement requires reasoning about the sets of objects in the image, comparing objects properties, and reasoning about spatial relations. The dataset is interesting for us for two reasons. First, the statements evaluation requires complex relational reasoning about the objects in the image. Second, unlike the bAbI dataset, the statements are written in natural language. Because of that, each statement displays a range of syntactic and semantic phenomena that are not present in the bAbI dataset.

Model details

Our model can be easily adapted to deal with visual information. Following the idea from Santoro et al. (2017), instead of processing each input using a recurrent neural network, we use a Convolutional Neural Network (CNN). The CNN takes as input each sub-image and convolved them through convolutional layers. The output of the CNN consists of k feature maps (where k is the number



One tower with one block block at the top | Answer:False / Pred: False

At least one square closely touching one box edge | Answer:True / Pred: True

Story (2 supporting facts)	Support	Hop 1	Hop 2	Hop 3	Hop 4
Mary moved to the office.		0.79	0.30	0.15	0.15
Sandra travelled to the bedroom.	True	0.02	2.64	2.75	0.39
Daniel dropped the football.		0.03	0.13	0.16	0.41
Sandra left the milk there.	True	1.01	0.07	0.16	0.38
Daniel grabbed the football there.		0.08	0.31	0.07	0.27
Question: Where is the milk? Answer: bedroom, Pred: bedroom					

Story (2 supporting facts)	Support	Hop 1	Hop 2	Hop 3	Hop 4
Brian is white.		0.46	0.36	0.35	0.89
Bernhard is white.		0.07	0.13	0.19	0.81
Julius is a frog.	True	0.16	2.03	0.39	0.26
Julius is white.	True	0.09	0.23	2.42	1.32
Greg is a frog.	True	1.95	1.60	0.77	0.25
Question: What color is greg? Answer: white, Pred: white					

Table 2: Examples of visualizations of attention for textual and visual QA. Top: Visualization of attention values for the NLVR dataset. To get more aesthetic figures we applied a gaussian blur to the attention matrix. Bottom: Attention values for the bAbI dataset. In each cell, the sum of the attention for all heads is shown.

of kernels in the final convolutional layer) of size $d \times d$. Then, each memory is built from the vector composed by the concatenation of the cells in the same position of each feature map. Consequently, $d \times d$ memories of size k are stored in the short-term storage. The statement is processed using a GRU neural network as in the textual reasoning task. Then, we can proceed using the same architecture for the reasoning and attention module that the one used in the textual QA model. However, for the visual QA task, we used an additive attention mechanism. The additive attention computes the attention weight using a feed-forward neural network applied to the concatenation of the memory vector and statement vector.

Results

Our model achieves a validation / test accuracy of $65.6\%/65.8\%$. Notably, we achieved a performance comparable to the results of the Module Neural Networks (Andreas et al., 2016) that make use of standard NLP tools to process the statements into structured representations. Unlike the Module Neural Networks, we achieved our results using only raw input statements, allowing the model to learn how to process the textual input by itself. Note that given the more complex nature of the language used in the NLVR dataset we needed to use a larger embedding size and GRU hidden layer than in the bAbI dataset (100 and 128 respectively). That, however, is a nice feature of separating the input from the reasoning and attention component: One way to process more complex language statements is increasing the capacity of the input module.

4.3 From $O(n^2)$ to $O(n)$

One of the major limitations of RNs is that they need to process each one of the memories in pairs. To do that, the RN must perform $O(n^2)$ forward and backward passes (where n is the number of memories). That becomes quickly prohibitive for a larger number of memories. In contrast, the dependence of the W-MemNN run times on the number of memories is linear. Note, however, that computation times in the W-MemNN depend quadratically on the size of the working memory buffer. Nonetheless, this number is expected to be much smaller than the number of memories. To compare both models we measured the wall-clock time for a forward and backward pass for a single batch of size 32. We performed these experiments on a GPU NVIDIA K80. Figure 2 shows the results.

4.4 Memory Visualizations

One nice feature from Memory Networks is that they allow some interpretability of the reasoning procedure by looking at the attention weights. At each hop, the attention weights show which parts of the memory the model found relevant to produce the output. RNs, on the contrary, lack of this feature. Table 2 shows the attention values for visual and textual question answering.

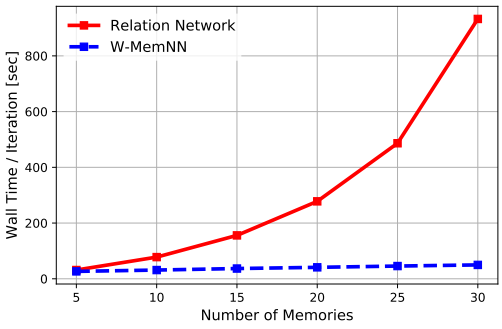


Figure 2: Wall-clock times for a forward and backward pass for a single batch. The batch size used is 32. While for 5 memories the times are comparable, for 30 memories the W-MemNN takes around 50s while the RN takes 930s, a speedup of almost 20×.

5 Conclusion

We have proposed a novel Working Memory Network architecture that introduces improved reasoning abilities to the original MemNN model. We demonstrated that by augmenting the MemNN architecture with a Relation Network, the computational complexity of the RN can be reduced, without loss of performance. That opens the opportunity for using RNs in larger problems, something that may be very useful, given the many tasks requiring a significant amount of memories.

Although we have used RN as the reasoning module in this work, other options can be tested. It might be interesting to analyze how other reasoning modules can improve different weaknesses of the model.

We presented results on the jointly trained bAbI-10k dataset, where we achieve a new state-of-the-art, with an average error of less than 0.5%. Also, we showed that our model can be easily adapted for visual question answering.

Our architecture combines perceptual input processing, short-term memory storage, an attention mechanism, and a reasoning module. While other models have focused on different parts of these components, we think that is important to find ways to combine these different mechanisms if we want to build models capable of complex reasoning. Evidence from cognitive sciences seems to show that all these abilities are needed in order to achieve human-level complex reasoning.

Acknowledgments

JP was supported by the Scientific and Technological Center of Valparaíso (CCTVal) under Fondecyt grant BASAL FB0821. HA was supported through the research project Fondecyt-Conicyt 1170123. The work of HAC was supported by the research project Fondecyt Initiation into Research 11150248.

References

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 39–48.

Alan Baddeley. 1992. Working memory. *Science* 255(5044):556–559.

Alan Baddeley. 2000. The episodic buffer: a new component of working memory? *Trends in cognitive sciences* 4(11):417–423.

Alan D Baddeley and Graham Hitch. 1974. Working memory. In *Psychology of learning and motivation*, Elsevier, volume 8, pages 47–89.

Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. 2016. Interaction networks for learning about objects, relations and physics. In *Advances in neural information processing systems*. pages 4502–4510.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* .

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. pages 249–256.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401* .

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. 2016. Hybrid computing using a neural network with dynamic external memory. *Nature* 538(7626):471.

Stevan Harnad. 1999. The symbol grounding problem. *CoRR* cs.AI/9906002. http://arxiv.org/abs/cs.AI/9906002.

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. 2016. Tracking the world state with recurrent entity networks. *arXiv preprint arXiv:1612.03969* .

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *International Conference on Machine Learning*. pages 1378–1387.

Moontae Lee, Xiaodong He, Wen-tau Yih, Jianfeng Gao, Li Deng, and Paul Smolensky. 2015. Reasoning in vector space: An exploratory study of question answering. *arXiv preprint arXiv:1511.06426* .

Fei Liu and Julien Perez. 2017. Gated end-to-end memory networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. volume 1, pages 1–10.

Rasmus Berg Palm, Ulrich Paquet, and Ole Winther. 2017. Recurrent relational networks for complex relational reasoning. *CoRR* abs/1711.08028. http://arxiv.org/abs/1711.08028.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*. pages 1310–1318.

Jack Rae, Jonathan J Hunt, Ivo Danihelka, Timothy Harley, Andrew W Senior, Gregory Wayne, Alex Graves, and Tim Lillicrap. 2016. Scaling memory-augmented neural networks with sparse reads and writes. In *Advances in Neural Information Processing Systems*. pages 3621–3629.

Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Tim Lillicrap. 2017. A simple neural network module for relational reasoning. In *Advances in neural information processing systems*. pages 4974–4983.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The graph neural network model. *IEEE Transactions on Neural Networks* 20(1):61–80.

Alane Suhr, Mike Lewis, James Yeh, and Yoav Artzi. 2017. A corpus of natural language for visual reasoning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, pages 217–223. https://doi.org/10.18653/v1/P17-2034.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*. pages 2440–2448.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. https://arxiv.org/pdf/1706.03762.pdf.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698* .

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *CoRR* abs/1410.3916. http://arxiv.org/abs/1410.3916.

Caiming Xiong, Stephen Merity, and Richard Socher. 2016. Dynamic memory networks for visual and textual question answering. In *International Conference on Machine Learning*. pages 2397–2406.