Recurrent Networks ← Relational → Learning

Reinforcement

Petar Veličković

Artificial Intelligence Group
Department of Computer Science and Technology, University of Cambridge, UK
In this talk, I will survey the Relational Network architecture, and its recent deployment in recurrent neural networks and deep reinforcement learning.

The discussion will span the following papers:

- A simple neural network module for relational reasoning (Santoro, Raposo et al., NIPS 2017)
- Relational deep reinforcement learning (Zambaldi, Raposo, Santoro et al., 2018)
- Relational recurrent neural networks (Santoro, Faulkner, Raposo et al., 2018)

Substantial part of DeepMind’s recent “graph networks surge”.
Relational reasoning

- Being able to reason about relations between entities present in an input is an important aspect of intelligence!

- Consider the simple task of inferring which two points from a given point set are furthest apart—this requires computing and comparing all* of their pairwise distances.

- Keep this task in mind—it will be revisited!
Approaches to relational reasoning

- Relations can be naturally expressed within *symbolic methods* (defined by e.g. the rules of logic)—but these are not robust to small variations of inputs/tasks.

- Robustness is often achievable with standard *neural network* architectures (such as MLPs), but it is extremely challenging for them to capture relations, despite their theoretical potency!
  - This claim is extensively validated throughout the three papers.

⇒ Seek a model inspired by *symbolic AI*, while empowered by *neural networks* (*explicitly* represent relations in a *robust* way).
Our task for today

How many outlined objects are above the spade?
Our task for today

How many outlined objects are above the spade?
Our task for today

"How many outlined objects are above the spade?"
Our task for today

"How many outlined objects are above the spade?"
The Relational Network

How many outlined objects are above the spade?
Initially, we will assume that the objects are provided as input.

Consider a set of \( n \) objects, \( \mathcal{O} = \{\vec{o}_1, \vec{o}_2, \ldots, \vec{o}_n\} \); with each object represented by a feature vector \( \vec{o}_i \in \mathbb{R}^m \).

A Relational Network (RN) summarises the relations between these objects as follows:

\[
RN(\mathcal{O}) = f_\phi \left( \sum_{i,j} g_\theta (\vec{o}_i, \vec{o}_j) \right)
\]

where \( g_\theta : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}^k \) and \( f_\phi : \mathbb{R}^k \rightarrow \mathbb{R}^l \) are functions with parameters \( \theta \) and \( \phi \) (usually MLPs).
Properties of RNs

- The central component of RNs is $g_\theta$; the relation function. Its role is to infer the nature of relations between objects $i$ and $j$.

- An RN may be seen as a message-passing neural network over the complete graph of object nodes.

- RNs have several highly desirable properties:
  - **Relational inference**—given the all-pairs nature of the computation, the module does not assume upfront knowledge of which pairs of objects are related, and how.
  - **Data efficiency**—an MLP would need to learn and embed $n^2$ (identical) functions to replicate behaviour of RNs.
  - **Permutation invariance**—the summation operation ensures that the order of objects does not matter; therefore RNs can be applied to arbitrary sets.
Dynamic physical systems

MuJoCo-simulated physical mass-spring systems with 10 objects.

Input: $\vec{o}_i$ is RGB color and $(x, y)$ coordinates across 16 time steps.

Tasks: (i) infer relations; (ii) count number of systems (harder!).
Results on physical systems

- Relational Networks achieve 93% accuracy in predicting the existence/absence of relations between objects, and 95% accuracy in predicting the number of interacting systems.
- MLPs fail to predict better than chance on either task!
- Learnt function transferable to unseen motion capture data!
Conditioning in RNs

"How many outlined objects are above the spade?"
An RN may be seen as a module that “captures” the relations between objects in a set—this computation may be arbitrarily conditioned, e.g. to answer a specific relational query.

Assuming we have a conditioning vector $\vec{q}$, the RN architecture may be trivially modified to include it:

$$RN(O, \vec{q}) = f_\phi \left( \sum_{i,j} g_\theta (\vec{o}_i, \vec{o}_j, \vec{q}) \right)$$
The CLEVR dataset (Johnson et al., 2017)

Question Answering dataset on 3D-rendered objects.

Original Image:

Non-relational question:
What is the size of the brown sphere?

Relational question:
Are there any rubber things that have the same size as the yellow metallic cylinder?

Input: $\tilde{o}_i$ is RGB color, $(x, y, z)$ coordinates, shape/material/size.

Queries: count, exist, compare numbers, query attribute, compare attribute.
The query sentence is encoded into $\tilde{q}$ as the last-stage output of a word-level LSTM (with learned word embeddings).

Relational Networks achieve an accuracy of 96.4% on CLEVR.

Human performance is 92.6%! This sounds great!
Results on CLEVR

- The query sentence is encoded into $\tilde{q}$ as the last-stage output of a word-level LSTM (with learned word embeddings).

- Relational Networks achieve an accuracy of **96.4%** on CLEVR.

- *Human performance is 92.6%! This sounds great!*

- *OK, I lied to you. (Sorry!)*
The actual CLEVR dataset (Johnson et al., 2017)

Visual Question Answering dataset on 3D-rendered objects.

Original Image:

Non-relational question:
What is the size of the brown sphere?

Relational question:
Are there any rubber things that have the same size as the yellow metallic cylinder?

Input: The scene image. The $\tilde{o}_i$ vectors are not explicitly given!
Queries: count, exist, compare numbers, query attribute, compare attribute.
"How many outlined objects are above the spade?"
In general, *we should not assume the $\tilde{\mathbf{o}}_i$ will be given!*

Arguably, obtaining the $\tilde{\mathbf{o}}_i$ from *raw input* will be the most variable pipeline component.

Often, we can obtain object representations as *high-level outputs* of neural networks specialised for such inputs.

In the case of images (*most common!*), this will be a *convolutional neural network*. 
A convolutional architecture generally consists of interleaving convolutional and pooling layers—progressively building more sophisticated feature maps.

At any point during a CNN, a feature map $f$ may have the shape $n \times m \times k$, where $n$ and $m$ are the height and width of the feature map, and each pixel is represented by $k$ features.

Each pixel represents a summary of a certain region of the image. Without any further assumptions, it is safest to let each pixel constitute an object!

Therefore, we will have an object set $\mathcal{O} = \{\vec{o}_1, \ldots \vec{o}_{n \cdot m}\}$ with $n \cdot m$ objects and $\vec{o}_i \in \mathbb{R}^k$ that will correspond to $\vec{f}_{xy}$. 
CNN object extraction

input → Conv. → Pool → Conv. → f
CNN object extraction

input

Conv.

Conv.

Pool

Conv.

⃗ o₁

⃗ o₃

⃗ o₂

⃗ o₄

⃗ o₁

⃗ o₂

⃗ o₃

⃗ o₄
Overall CLEVR architecture

End-to-end trainable with gradient descent.
Actual results on CLEVR

<table>
<thead>
<tr>
<th>Model</th>
<th>Overall</th>
<th>Count</th>
<th>Exist</th>
<th>Compare Numbers</th>
<th>Query Attribute</th>
<th>Compare Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human</td>
<td>92.6</td>
<td>86.7</td>
<td>96.6</td>
<td>86.5</td>
<td>95.0</td>
<td>96.0</td>
</tr>
<tr>
<td>Q-type baseline</td>
<td>41.8</td>
<td>34.6</td>
<td>50.2</td>
<td>51.0</td>
<td>36.0</td>
<td>51.3</td>
</tr>
<tr>
<td>LSTM</td>
<td>46.8</td>
<td>41.7</td>
<td>61.1</td>
<td>69.8</td>
<td>36.8</td>
<td>51.8</td>
</tr>
<tr>
<td>CNN+LSTM</td>
<td>52.3</td>
<td>43.7</td>
<td>65.2</td>
<td>67.1</td>
<td>49.3</td>
<td>53.0</td>
</tr>
<tr>
<td>CNN+LSTM+SA</td>
<td>68.5</td>
<td>52.2</td>
<td>71.1</td>
<td>73.5</td>
<td>85.3</td>
<td>52.3</td>
</tr>
<tr>
<td>CNN+LSTM+SA*</td>
<td>76.6</td>
<td>64.4</td>
<td>82.7</td>
<td>77.4</td>
<td>82.6</td>
<td>75.4</td>
</tr>
<tr>
<td>CNN+LSTM+RN</td>
<td><strong>95.5</strong></td>
<td><strong>90.1</strong></td>
<td><strong>97.8</strong></td>
<td><strong>93.6</strong></td>
<td><strong>97.9</strong></td>
<td><strong>97.1</strong></td>
</tr>
</tbody>
</table>

First approach to achieve *superhuman performance* on this task!
Actual results on CLEVR

Especially excels at compare attribute, the query type which heavily relies on relational reasoning.
Failure cases on CLEVR

Failure inputs are often occurring under heavy occlusion—challenging for humans as well!

<table>
<thead>
<tr>
<th>Question</th>
<th>RN:</th>
<th>GT:</th>
</tr>
</thead>
<tbody>
<tr>
<td>What shape is the small object that is in front of the yellow matte thing and behind the gray sphere?</td>
<td>cylinder</td>
<td>cube</td>
</tr>
<tr>
<td>What number of things are either tiny green rubber objects or shiny things that are behind the big metal block?</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>What number of objects are blocks that are in front of the large red cube or green balls?</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>
Results on Sort-of-CLEVR

A simple CLEVR-inspired dataset with clear separation of relational vs. non-relational queries.

Non-relational question
Q: What is the shape of the gray object?
A: circle

Relational question
Q: What is the shape of the object that is furthest from the gray object?
A: square

Demonstrates clear advantage of RNs on relational queries.
LSTM object extraction: bAbI (Weston et al., 2015)

A text-based set of 20 question-answering tasks.

Let $O = \{\vec{o}_1, \ldots, \vec{o}_{20}\}$ be the LSTM representations of up to 20 sentences preceding the question. $\vec{q}$ is once again obtained as the LSTM representation of the question.
Results on bAbI

- $RN(\emptyset, \tilde{q})$ passes (95+% accuracy) **18/20 tasks** after joint training—comparable with other state-of-the-art memory network architectures.
  - Memory networks: 14/20
  - DNC: 18/20
  - Sparse DNC: 19/20
  - EntNet: 16/20
- Does not **catastrophically fail** (91.9% and 83.5% accuracy) on the remaining two.
- Notably, it succeeds on the **basic induction** task (97.9%), where Sparse DNC (46%), DNC (44.9%) and EntNet (47.9%) all fail.
First, the building block functions of a Relational Network $(f_\phi, g_\theta)$ were simple MLPs.

For more recent RN architectures, we focus instead on the self-attention operator.

A self-attentional operator, $A_\theta$, acts on a set of $n$ entities, $\mathcal{E} = \{\vec{e}_1, \vec{e}_2, \ldots, \vec{e}_n\}$, producing higher-level representations:

$$\tilde{\mathcal{E}} = A_\theta(\mathcal{E})$$

where $\tilde{\mathcal{E}} = \{\vec{e}'_1, \vec{e}'_2, \ldots, \vec{e}'_n\}$, and $\theta$ are learnable parameters.
Each component of $\tilde{E}$ will be derived by examining all components of $E$ (by way of linear combinations):

$$
\tilde{e}_i' = \sum_j \alpha_{ij} f_\psi(\tilde{e}_j)
$$

where $f_\psi : \mathbb{R}^m \rightarrow \mathbb{R}^k$ is a learnable transformation.

Here, the coefficients $\alpha_{ij}$ correspond to the importance of the features of entity $j$ to entity $i$, and are derived by a learnable attention mechanism, $a_\phi : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}$:

$$
\alpha_{ij} = a_\phi(\tilde{e}_i, \tilde{e}_j)
$$
Self-attention
In particular, the Transformer architecture (Vaswani et al., 2017) is used for $A_\theta$.

- Here abbreviated as MHDPA (multi-head dot-product attention).

First, derive queries, keys and values for the attention:

$$\vec{q}_i = W_q \vec{e}_i \quad \vec{k}_i = W_k \vec{e}_i \quad \vec{v}_i = W_v \vec{e}_i$$

Now, use the queries and keys to derive coefficients:

$$\alpha_{ij} = \frac{\exp \left( \langle \vec{q}_i, \vec{k}_j \rangle / \sqrt{d_k} \right)}{\sum_m \exp \left( \langle \vec{q}_i, \vec{k}_m \rangle / \sqrt{d_k} \right)}$$

where $d_k$ is the dimensionality of the keys.
The *Transformer* architecture, *cont’d*

- Now, can use $\alpha_{ij}$ to **recombine** the values at each position:
  \[ \vec{e}'_i = \sum_j \alpha_{ij} \vec{v}_j \]

- Can be conveniently written in matrix form as:
  \[ \tilde{E} = \text{softmax} \left( \frac{QK^T}{\sqrt{d_k}} \right) V \]

- Further optimised by using **multi-head attention**; replicating this operation $K$ times (each with independent parameters $W_q$, $W_k$, $W_v$) and featurewise-concatenating the results.
**Box-World:** A grid RL environment meant to stress relational reasoning while deciding how to act.

**StarCraft II:** Mini-games (Vinyals et al., 2017).

In both cases, agent receives pixel-structured inputs (minimaps, screens, etc.).
Relational deep reinforcement learning

- Empowers a standard CNN-based policy network (in an RL setting) with a relational module based on self-attention.
  - Architectures for both tasks are very similar!

- Extract entities, $\tilde{e}_i$, just as before (as separate pixels in a high-level feature map).

- Then perform several rounds of the Transformer self-attention over $\mathcal{E}$ (each round followed by a small MLP, $f_\theta$, and layer normalisation to introduce nonlinearity).

- Finally, perform global pooling and a small MLP to derive the policy for the RL algorithm (IMPALA (Espeholt et al., 2018)).
The Box-World architecture

ReLU

FC 256

Relational
module

x 2

x 4

Feature-wise
max pooling

Conv. 2 x 2, stride 1

Multi-head dot product attention

ReLU

ReLU

FC 256

Relational
module

x 2

x 4

Input

Feature-wise
max pooling

query $q_i$

key $k_i$

value $v_i$

$E$

$A$

$\tilde{E}$

softmax($\frac{QK^T}{\sqrt{d}}$)$V$
Results on Box-World

Observation

Underlying graph

Branch length = 1

Branch length = 3

Environment steps

Fraction solved

Relational (1 block)
Relational (2 blocks)
Baseline (3 blocks)
Baseline (6 blocks)
Relational (2 blocks)
Relational (4 blocks)
Visualising the attentional coefficients

a) Underlying graph

b) Entity 1 | Entity 2 | Entity 3 | Entity 4 | Entity 5

Attention head 1

Attention head 2
Zero-shot experiments in Box-World

a) Longer solution path lengths

b) Withheld key

The non-relational baseline (ResNet CNN) **fails** to generalise!
Table 1: Mean scores achieved in the StarCraft II mini-games using full action set. ↑ denotes a score that is higher than a StarCraft Grandmaster. Mini-games: (1) Move To Beacon, (2) Collect Mineral Shards, (3) Find And Defeat Zerglings, (4) Defeat Roaches, (5) Defeat Zerglings And Banelings, (6) Collect Minerals And Gas, (7) Build Marines.

<table>
<thead>
<tr>
<th>Agent</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Policy [15]</td>
<td>1</td>
<td>17</td>
<td>4</td>
<td>1</td>
<td>23</td>
<td>12</td>
<td>&lt;1</td>
</tr>
<tr>
<td>FullyConv LSTM [15]</td>
<td>26</td>
<td>104</td>
<td>44</td>
<td>98</td>
<td>96</td>
<td>3351</td>
<td>6</td>
</tr>
<tr>
<td>PBT-A3C [33]</td>
<td>–</td>
<td>101</td>
<td>50</td>
<td>132</td>
<td>125</td>
<td>3345</td>
<td>0</td>
</tr>
<tr>
<td>Relational agent</td>
<td>27</td>
<td>196</td>
<td>62</td>
<td>303</td>
<td>736</td>
<td>4906</td>
<td>123</td>
</tr>
<tr>
<td>Control agent</td>
<td>27</td>
<td>187</td>
<td>61</td>
<td>295</td>
<td>602</td>
<td>5055</td>
<td>120</td>
</tr>
</tbody>
</table>

Sets new **state-of-the-art**, often **beating human grandmaster**.
Zero-shot experiments in StarCraft

Exhibits higher—although not fully conclusive—generalisation ability from 2 marines to higher numbers.
Finally, we turn our attention to architectures used for general **sequential processing** of data.

In the general setting, we require a *stateful system*, $S_\theta$, capable of processing incoming inputs $\vec{x}_t$, and updating its internal state, $\vec{s}_t$, appropriately:

$$\vec{s}_t = S_\theta(\vec{x}_t, \vec{s}_{t-1})$$

Inference may then be performed by leveraging $\vec{s}_t$. 
Traditional approaches for modelling $S_\theta$ include recurrent neural networks (e.g. LSTM, GRU, etc.) and memory-augmented neural networks (e.g. NTM, DNC, etc.).

Recurrent neural networks generally represent their state as a fixed-size vector, $\tilde{c}_t$, which gets appropriately updated at each stage of input processing.

Memory-augmented networks have a memory matrix, $\mathcal{M} \in \mathbb{R}^{n \times m}$, which may be read from/written to by using a recurrent controller.
Both approaches have shortcomings when explicit relational reasoning through time is required:

- RNNs pack entire representation in a single dense vector, making it hard to reason about entities (and therefore relations);
- Memory-augmented networks explicitly represent entities (as rows of $\mathcal{M}$), but these cannot easily interact once written to.

Relational recurrent neural networks address both shortcomings simultaneously, explicitly allowing rows of $\mathcal{M}$ to interact using self-attention!
Assume we have a memory matrix $\mathcal{M} = \{\vec{m}_1, \vec{m}_2, \ldots, \vec{m}_n\}$.

Applying (Transformer) self-attention to it, we obtain a new memory state $\tilde{\mathcal{M}} = \{\vec{m}_1', \vec{m}_2', \ldots, \vec{m}_n'\}$, explicitly taking into account the relations between memory rows $\vec{m}_i$:

$$\tilde{\mathcal{M}} = \text{softmax} \left( \frac{QK^T}{\sqrt{d_k}} \right) V$$

where

$$Q = \mathcal{M} W_q \quad K = \mathcal{M} W_k \quad V = \mathcal{M} W_v$$

just as before.
Incorporating new inputs

- The interactions thus far are *self-contained* to what’s already in the memory; however, we’d like the memory to adapt to *incoming inputs*, $\vec{x}$, appropriately.

- Simple extension: let the memory locations attend over $\vec{x}$ too!

$$
\widetilde{\mathcal{M}} = \text{softmax} \left( \frac{Q[\|W_k\vec{x}]^T}{\sqrt{d_k}} \right) [V\|W_v\vec{x}] 
$$

where $\|$ denotes row-concatenation.
We do not wish to fully overwrite $\mathcal{M}$ by $\widetilde{\mathcal{M}}$—can control this process with an LSTM:

$$\vec{i}_{i,t} = \sigma \left( W_i \vec{x}_t + U_i \vec{h}_{i,t-1} + \vec{b}_i \right)$$

$$\vec{f}_{i,t} = \sigma \left( W_f \vec{x}_t + U_f \vec{h}_{i,t-1} + \vec{b}_f \right)$$

$$\vec{o}_{i,t} = \sigma \left( W_o \vec{x}_t + U_o \vec{h}_{i,t-1} + \vec{b}_o \right)$$

$$\vec{m}_{i,t} = g_\psi (\vec{m}'_{i,t}) \odot \vec{i}_{i,t} + \vec{m}_{i,t-1} \odot \vec{f}_{i,t}$$

$$\vec{h}_{i,t} = \text{tanh} \left( \vec{m}_{i,t} \odot \vec{o}_{i,t} \right)$$

where $g_\psi$ is a learnable function (2-layer MLP with layer normalisation in the paper).
The Relational RNN

(a) CORE

(b) MULTI-HEAD DOT PRODUCT ATTENTION

(c) Computation steps:
- Compute attention weights
- Normalize weights with row-wise softmax
- Compute weighted average of values
- Return updated memory

### Core Components
- **Input**
- **Residual**
- **MLP**
- **Apply gating**
- **Output**
- **Prev. Memory**
- **Next Memory**
- **Computation of gates not depicted**

### Multi-Head Dot Product Attention
- **Queries**
- **Keys**
- **Values**
- **Weights**
- **Normalized Weights**
- **Updated Memory**

### Attention Computation
- \( QK^T \)
- \( \text{softmax}(QK^T) \)
- \( \text{softmax}(QK^T)V \)
- \( \tilde{M} \)
- \( m_{4,4} \)
Tasks under consideration

A suite of supervised and reinforcement learning tasks demanding explicit sequential relational reasoning.

What is the $N$th farthest from vector $m$?

```python
x = 339
for [19]:
    x += 597
        for [94]:
  x += 875
x if 428 < 778 else 652
print(x)
```

Super Mario Land is a 1989 side scrolling platform video ______

It had 24 step programming abilities, which meant it was highly ______

A gold dollar had been proposed several times in the 1830s and 1840s, but was not initially ______

Nth farthest Program Evaluation Language Modeling

Supervised Learning

BoxWorld Mini-Pacman

Reinforcement Learning

- $N$-th farthest vector from a given vector;
- Program evaluation from characters (*Learning to Execute*);
- Language modelling;
- Mini-PacMan and Box-World *with viewport!*
Results on $N$-th farthest: LSTM/DNC

Failing to surpass 30% batch accuracy!
Results on $N$-th farthest: RRNN
Attention weight visualisation

(a) Reference vector is the last in a sequence, e.g. "Choose the 5th furthest vector from vector 7"

(b) Reference vector is the first in a sequence, e.g. "Choose the 3rd furthest vector from vector 4"

(c) Reference vector comes in the middle of a sequence, e.g. "Choose the 6th furthest vector from vector 6"
Results on LTE

The RRNN is again highly competitive, especially in scenarios where strong relational reasoning may be required (full programs).

<table>
<thead>
<tr>
<th>Model</th>
<th>Add</th>
<th>Control</th>
<th>Program</th>
<th>Copy</th>
<th>Reverse</th>
<th>Double</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM [3, 37]</td>
<td>99.8</td>
<td>97.4</td>
<td>66.1</td>
<td>99.8</td>
<td>99.7</td>
<td>99.7</td>
</tr>
<tr>
<td>EntNet [38]</td>
<td>98.4</td>
<td>98.0</td>
<td>73.4</td>
<td>91.8</td>
<td>100.0</td>
<td>62.3</td>
</tr>
<tr>
<td>DNC [5]</td>
<td>99.4</td>
<td>83.8</td>
<td>69.5</td>
<td>100.0</td>
<td>100.0</td>
<td>100.0</td>
</tr>
<tr>
<td>Relational Memory Core</td>
<td>99.9</td>
<td>99.6</td>
<td>79.0</td>
<td>100.0</td>
<td>100.0</td>
<td>99.8</td>
</tr>
</tbody>
</table>
Results on LTE
Results on Language Modelling

Table 2: Validation and test perplexities on WikiText-103, Project Gutenberg, and GigaWord v5.

<table>
<thead>
<tr>
<th></th>
<th>WikiText-103</th>
<th></th>
<th>Gutenberg</th>
<th></th>
<th>GigaWord</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Valid Test</td>
<td></td>
<td>Valid Test</td>
<td></td>
<td>Test</td>
</tr>
<tr>
<td>LSTM [40]</td>
<td>-</td>
<td>48.7</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Temporal CNN [41]</td>
<td>-</td>
<td>45.2</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Gated CNN [42]</td>
<td>-</td>
<td>37.2</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>LSTM [32]</td>
<td>34.1</td>
<td>34.3</td>
<td>41.8</td>
<td>45.5</td>
<td>43.7</td>
</tr>
<tr>
<td>Quasi-RNN [43]</td>
<td>32</td>
<td>33</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Relational Memory Core</td>
<td><strong>30.8</strong></td>
<td><strong>31.6</strong></td>
<td><strong>39.2</strong></td>
<td><strong>42.0</strong></td>
<td><strong>38.3</strong></td>
</tr>
</tbody>
</table>

The RRNN obtains competitive perplexity levels, compared to several strong baselines.
Results on Language Modelling: WikiText-103

![Graph showing the performance of RMC and LSTM models over steps (B). The graph plots Perplexity on the y-axis against Steps (B) on the x-axis. The RMC model shows a faster decline in perplexity compared to the LSTM model.]
Results on Mini-PacMan

The RRNN outperforms an LSTM when used as a policy network (for IMPALA). Specifically, when the entire map is observed, it **doubles** the LSTM performance!
Empowering neural networks with various kinds of relational reasoning modules will likely be a necessary step towards strong and robust intelligent systems.

- This claim is clearly supported by several “failure modes” of baseline architectures we considered today.

- One limitation going forward lies in the all-pairs interactions, which will limit scalability to larger object sets, especially if self-attention is used.
  - The NRI (Kipf, Fetaya et al., 2018) offers one possible direction to address this, but probably not the ultimate solution.

- In my opinion, particularly important avenue for future work are graph-structured memories; where we are not restricted to a matrix, and relations between slots are not all-pairs.
Questions?

petar.velickovic@cst.cam.ac.uk

http://www.cst.cam.ac.uk/~pv273/