UNIVERSITY OF CAMBRIDGE

$$
\begin{array}{ccccc}
 & & \circlearrowleft & & \\
 & & \textbf{Recurrent} & & \\
 & \swarrow & \uparrow & & \\
\textbf{Networks} & \leftarrow & \textbf{Relational} & \rightarrow & \textbf{Learning} \\
 & & \downarrow & \nearrow & \\
 & & \textbf{Reinforcement} & &
\end{array}
$$

Petar $\rightsquigarrow$ Veličković

Artificial Intelligence Group
Department of Computer Science and Technology, University of Cambridge, UK

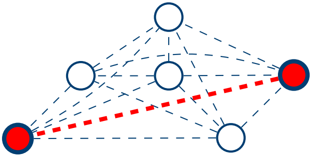
MILA Graph Representation Reading Group Meeting 22 August 2018

Introduction

- 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!

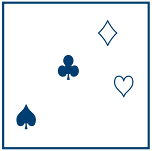
Approaches to relational reasoning

- 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.

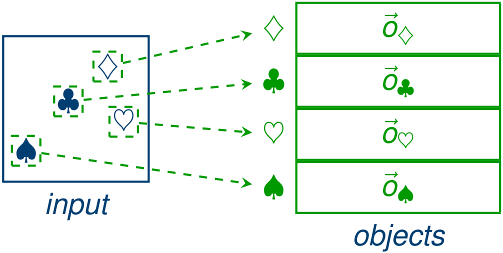
$\Longrightarrow$ Seek a model inspired by *symbolic AI*, while empowered by *neural networks* (*explicitly* represent relations in a *robust* way).

Our task for today

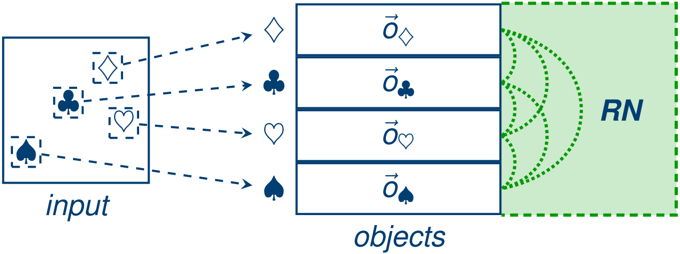


input

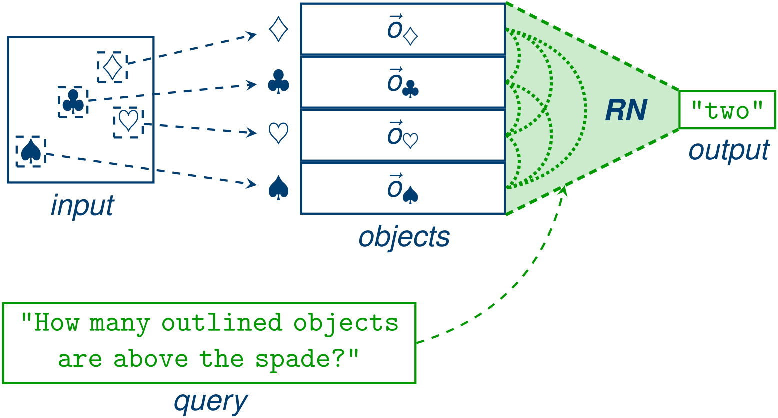
Our task for today



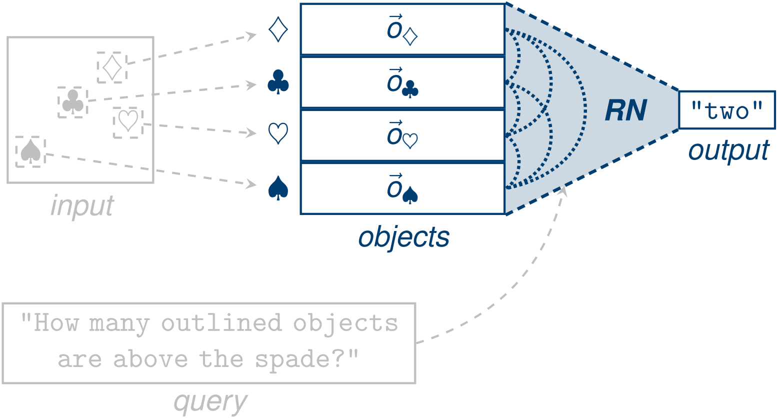
Our task for today



Our task for today



The Relational Network



Relational Networks

- 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:

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

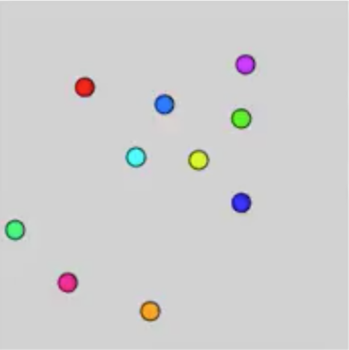
where $g_\theta : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}^k$ and $f_\phi : \mathbb{R}^k \rightarrow \mathbb{R}^l$ are functions with parameters θ and ϕ (usually MLPs).

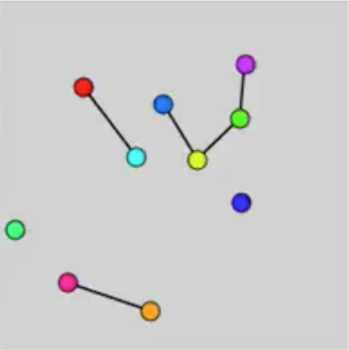
Properties of RNs

- The central component of RNs is g_θ; 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.

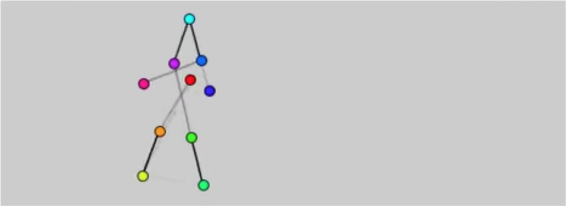




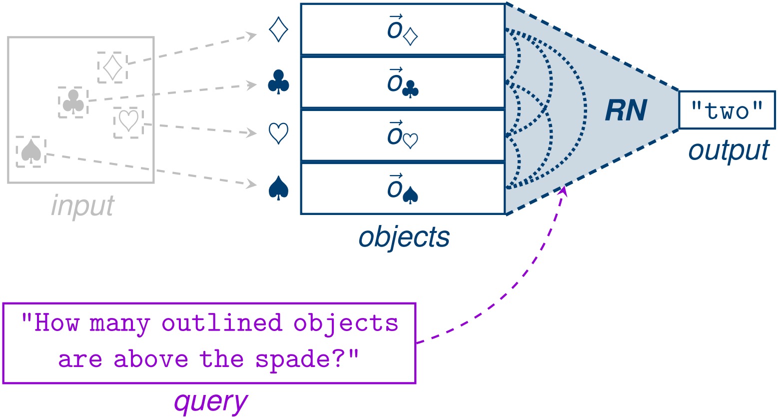
Input: $\vec{o}_i$ is RGB color and (x, y) coordinates across 16 time steps.
Tasks: (i) infer relations; (ii) count number of systems (harder!).

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



RN conditioning

- 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.

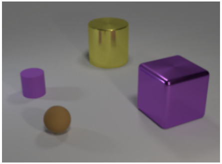
- Assuming we have a conditioning vector $\vec{q}$, the RN architecture may be trivially modified to include it:

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

The CLEVR dataset (Johnson *et al.*, 2017)

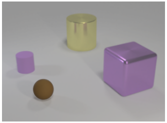
Question Answering dataset on 3D-rendered objects.

Original Image:



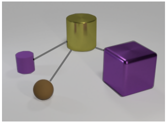
Non-relational question:

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

Queries: `count`, `exist`, `compare numbers`, `query attribute`, `compare attribute`.

Results on CLEVR

- The query sentence is encoded into $\vec{q}$ 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!
- ...

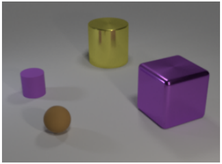
Results on CLEVR

- The query sentence is encoded into $\vec{q}$ 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)

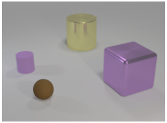
Visual Question Answering dataset on 3D-rendered objects.

Original Image:



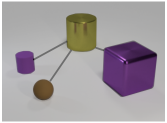
Non-relational question:

```
What is the size of
the brown sphere?
```



Relational question:

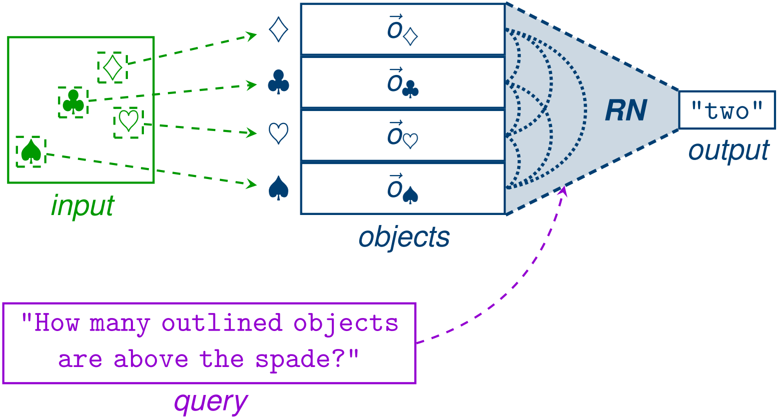
```
Are there any rubber
things that have the
same size as the yellow
metallic cylinder?
```



Input: **The scene image.** *The $\vec{o}_i$ vectors are **not** explicitly given!*

Queries: `count`, `exist`, `compare numbers`, `query attribute`, `compare attribute`.

Object extraction



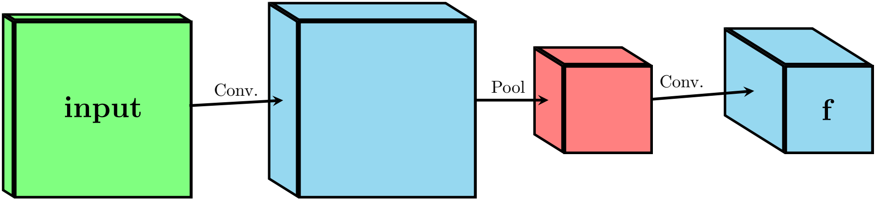
Object extraction from *images*

- In general, *we should not assume the $\vec{o}_i$ will be given!*
- Arguably, obtaining the $\vec{o}_i$ 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*.

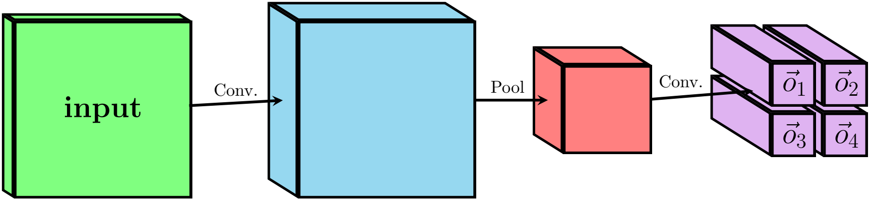
CNN object extraction

- 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 $\mathbf{f}$ 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 $\mathcal{O} = \{\vec{o}_1, \dots \vec{o}_{n \cdot m}\}$ with $n \cdot m$ objects and $\vec{o}_i \in \mathbb{R}^k$ that will correspond to $\vec{f}_{xy}$.

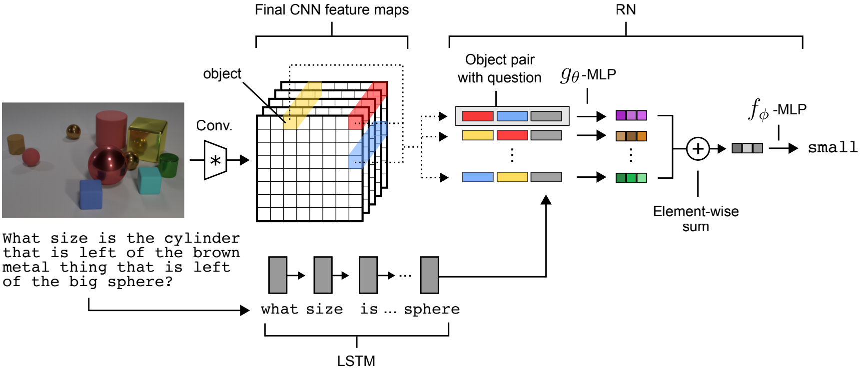
CNN object extraction



CNN object extraction



Overall CLEVR architecture



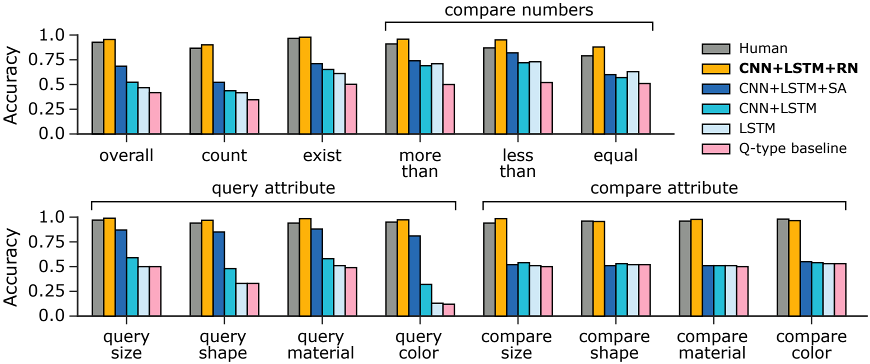
End-to-end trainable with gradient descent.

Actual results on CLEVR

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

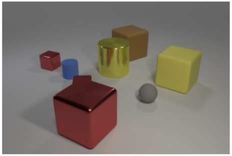
First approach to achieve *superhuman performance* on this task!

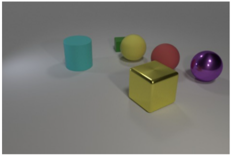
Actual results on CLEVR

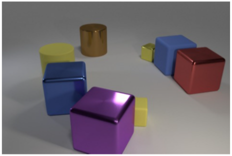


Especially excels at `compare attribute`, the query type which heavily relies on relational reasoning.

Failure cases on CLEVR







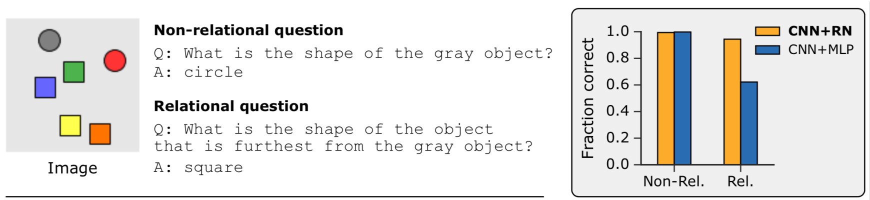
	What shape is the small object that is in front of the yellow matte thing and behind the gray sphere?	What number of things are either tiny green rubber objects or shiny things that are behind the big metal block?	What number of objects are blocks that are in front of the large red cube or green balls?
RN:	cylinder	1	2
GT:	cube	2	3

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

UNIVERSITY OF CAMBRIDGE

Results on Sort-of-CLEVR

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



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.

Task 1: Single Supporting Fact
Mary went to the bathroom.
John moved to the hallway.
Mary travelled to the office.
Where is Mary? A:office

Task 2: Two Supporting Facts
John is in the playground.
John picked up the football.
Bob went to the kitchen.
Where is the football? A:playground

Let $\mathcal{O} = \{\vec{o}_1, \ldots, \vec{o}_{20}\}$ 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*$(\mathcal{O}, \vec{q})$ 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.

Self-attention

- Thus far, the building block functions of a Relational Network (f_ϕ, g_θ) were simple MLPs.
- For more recent RN architectures, we focus instead on the *self-attention* operator.
- A self-attentional operator, A_θ, acts on a set of n entities, $\mathcal{E} = \{\vec{e}_1, \vec{e}_2, \dots, \vec{e}_n\}$, producing higher-level representations:

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

where $\widetilde{\mathcal{E}} = \{\vec{e}'_1, \vec{e}'_2, \dots, \vec{e}'_n\}$, and θ are learnable parameters.

Self-attention, *cont'd*

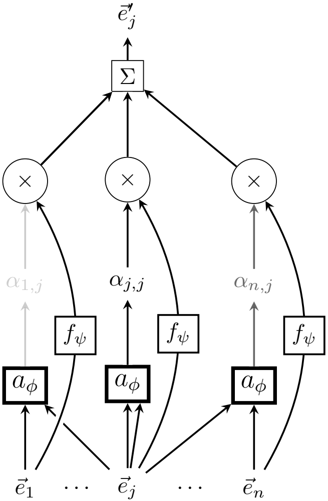
- Each component of $\widetilde{\mathcal{E}}$ will be derived by examining all components of $\mathcal{E}$ (by way of linear combinations):

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

  where $f_\psi : \mathbb{R}^m \rightarrow \mathbb{R}^k$ is a learnable transformation.
- Here, the coefficients α_{ij} correspond to the *importance* of the features of entity j to entity i, and are derived by a learnable *attention mechanism*, $a_\phi : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}$:

$$\alpha_{ij} = a_\phi(\vec{e}_i, \vec{e}_j)$$

Self-attention



The *Transformer* architecture

- In particular, the **Transformer** architecture (Vaswani *et al.*, 2017) is used for A_θ.
  - Here abbreviated as **MHDPA** (*multi-head dot-product attention*).
- First, derive *queries, keys and values* for the attention:

$$\vec{q}_i = \mathbf{W}_q \vec{e}_i \qquad \vec{k}_i = \mathbf{W}_k \vec{e}_i \qquad \vec{v}_i = \mathbf{W}_v \vec{e}_i$$

- 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 α_{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:

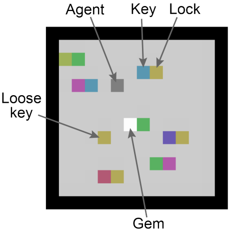
$$\widetilde{\mathcal{E}} = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}$$

- Further optimised by using **multi-head attention**; replicating this operation K times (each with independent parameters $\mathbf{W}_q$, $\mathbf{W}_k$, $\mathbf{W}_v$) and featurewise-concatenating the results.

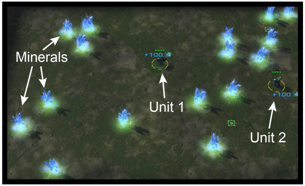
Reinforcement learning: *Box-World* and *StarCraft II*

Box-World: A grid RL environment meant to stress relational reasoning while deciding how to act.
StarCraft II: Mini-games (Vinyals *et al.*, 2017).

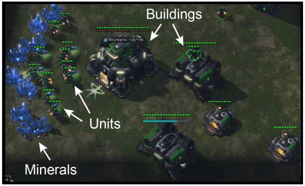
Box-World



StarCraft II mini-game
Collect Mineral Shards



StarCraft II mini-game
Build Marines

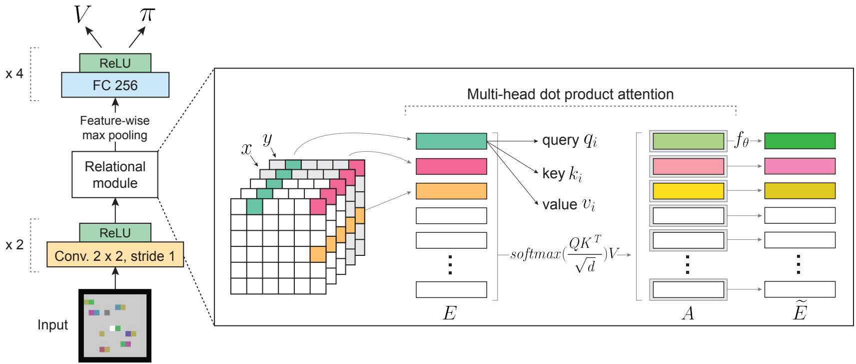


In both cases, agent receives pixel-structured inputs (minimaps, screens, etc.).

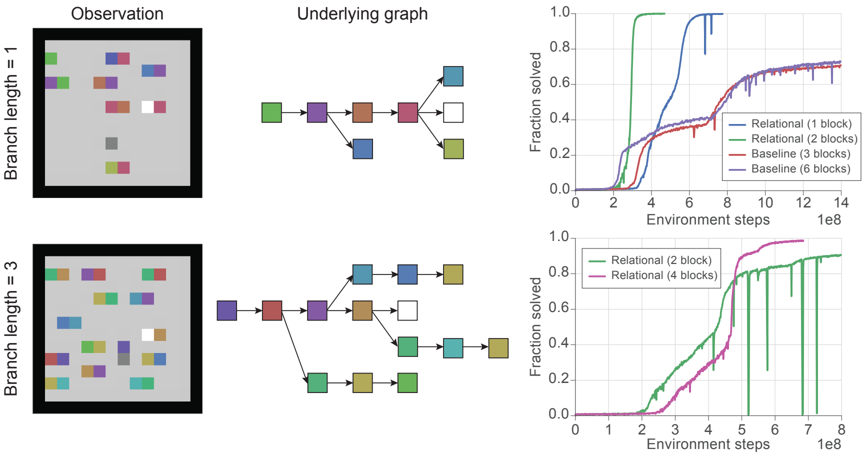
Relational deep reinforcement learning

- 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, $\vec{e}_i$, 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_θ, 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)).

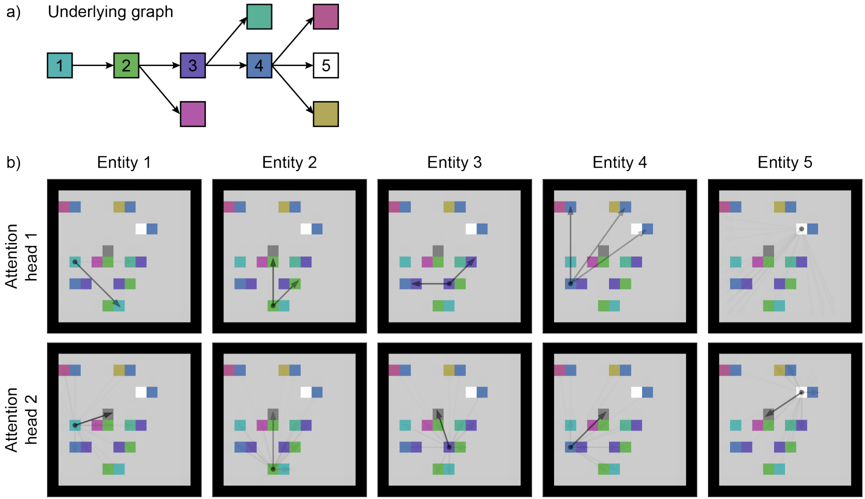
The Box-World architecture



Results on Box-World

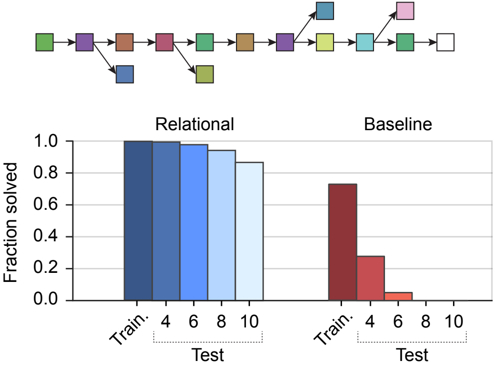


Visualising the attentional coefficients

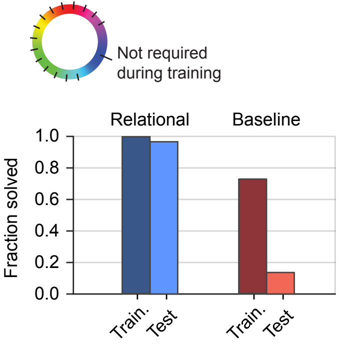


Zero-shot experiments in Box-World

a) Longer solution path lengths



b) Withheld key



The non-relational baseline (ResNet CNN) **fails** to generalise!

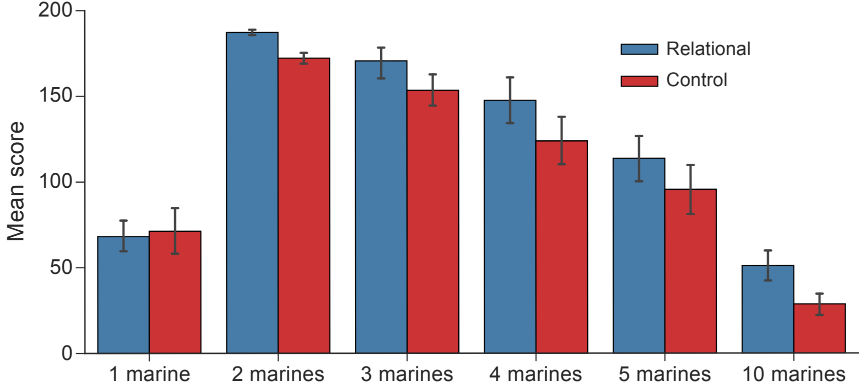
Results on StarCraft II

Agent	Mini-game 1	2	3	4	5	6	7
DeepMind Human Player [15]	26	133	46	41	729	6880	138
StarCraft Grandmaster [15]	28	177	61	215	727	7566	133
Random Policy [15]	1	17	4	1	23	12	< 1
FullyConv LSTM [15]	26	104	44	98	96	3351	6
PBT-A3C [33]	–	101	50	132	125	3345	0
Relational agent	**27**	**196** ↑	**62** ↑	**303** ↑	**736** ↑	4906	**123**
Control agent	**27**	187 ↑	61	295 ↑	602	**5055**	120

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.

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.

Neural networks for sequential processing

- Finally, we turn our attention to architectures used for general **sequential processing** of data.

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

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

- Inference may then be performed by leveraging $\vec{s}_t$.

Approaches to sequential processing

- Traditional approaches for modelling $\mathcal{S}_\theta$ 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, $\vec{c}_t$, 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*.

Analysis of approaches

- 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*!

Relational memory

- 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 $\widetilde{\mathcal{M}} = \{\vec{m}'_1, \vec{m}'_2, \ldots, \vec{m}'_n\}$, explicitly taking into account the relations between memory rows $\vec{m}_i$:

$$\widetilde{\mathcal{M}} = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}$$

where

$$\mathbf{Q} = \mathcal{M}\mathbf{W}_q \qquad \mathbf{K} = \mathcal{M}\mathbf{W}_k \qquad \mathbf{V} = \mathcal{M}\mathbf{W}_v$$

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!

$$\widetilde{\mathcal{M}} = \text{softmax}\left(\frac{\mathbf{Q}[\mathbf{K}\|\mathbf{W}_k\vec{x}]^T}{\sqrt{d_k}}\right)[\mathbf{V}\|\mathbf{W}_v\vec{x}]$$

where $\|$ denotes row-concatenation.

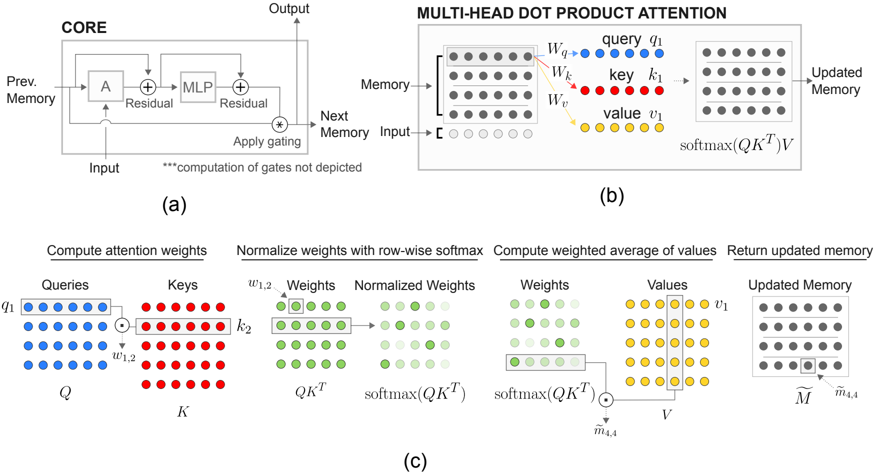
The LSTM controller

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

$$
\begin{aligned}
\vec{i}_{i,t} &= \sigma\left(\mathbf{W}_i \vec{x}_t + \mathbf{U}_i \vec{h}_{i,t-1} + \vec{b}_i\right) \\
\vec{f}_{i,t} &= \sigma\left(\mathbf{W}_f \vec{x}_t + \mathbf{U}_f \vec{h}_{i,t-1} + \vec{b}_f\right) \\
\vec{o}_{i,t} &= \sigma\left(\mathbf{W}_o \vec{x}_t + \mathbf{U}_o \vec{h}_{i,t-1} + \vec{b}_o\right) \\
\vec{m}_{i,t} &= g_\psi(\vec{m}'_{i,t}) \odot \vec{i}_{i,t} + \vec{m}_{i,t-1} \odot \vec{f}_{i,t} \\
\vec{h}_{i,t} &= \tanh\left(\vec{m}_{i,t}\right) \odot \vec{o}_{i,t}
\end{aligned}
$$

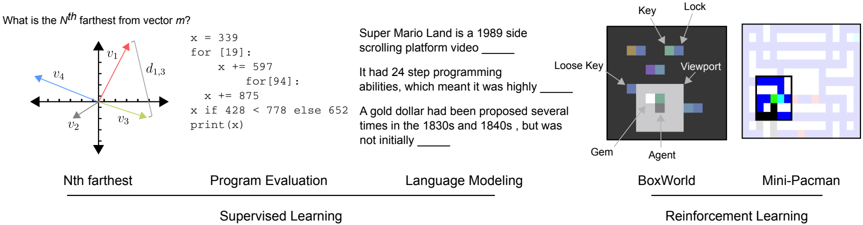
where g_ψ is a learnable function (2-layer MLP with layer normalisation in the paper).

The Relational RNN



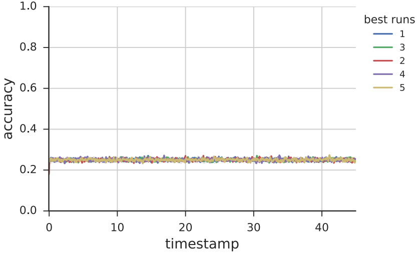
Tasks under consideration

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

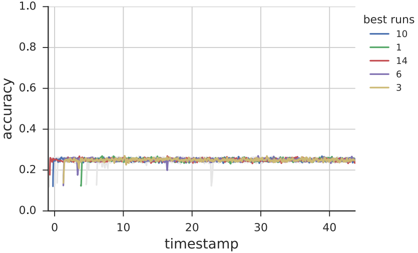


- *N*-th farthest vector from a given vector;
- Program evaluation from characters (*Learning to Execute*);
- Language modelling;
- Mini-PacMan and Box-World *with viewport*!

Results on N-th farthest: LSTM/DNC



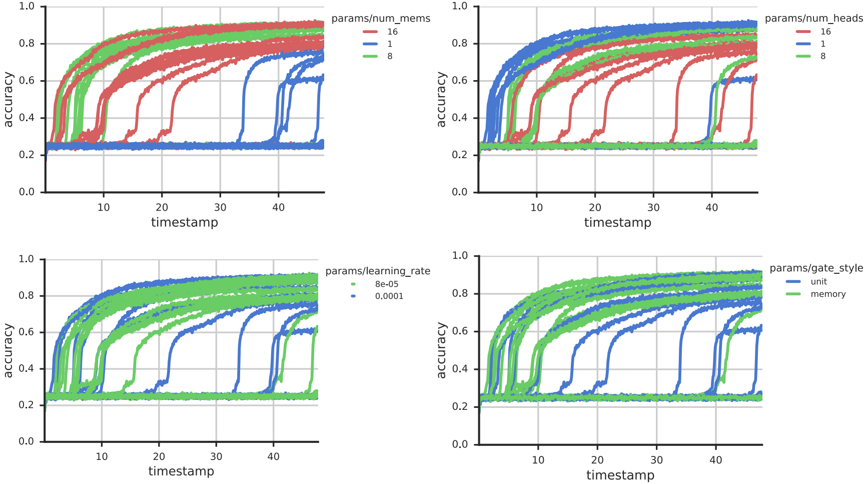
LSTM



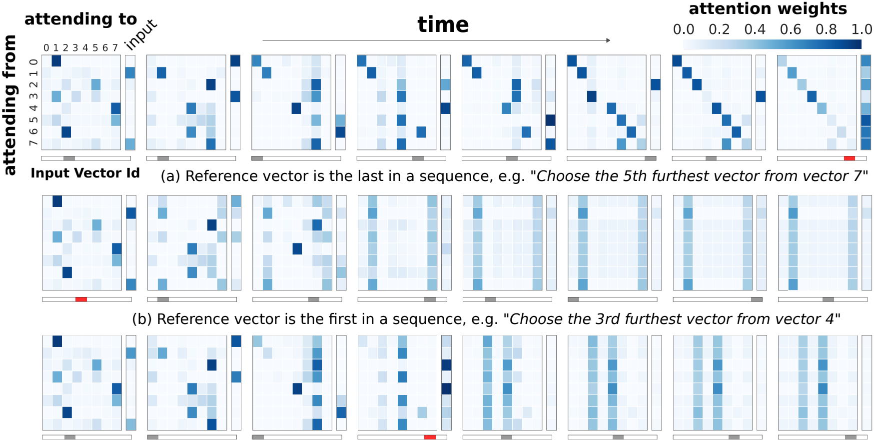
DNC

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*"

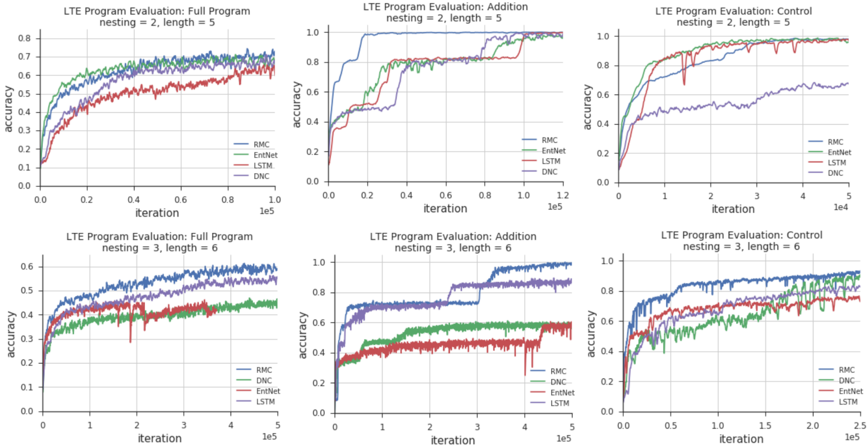
Results on LTE

Table 1: Test per character Accuracy on Program Evaluation and Memorization tasks.

Model	Add	Control	Program	Copy	Reverse	Double
LSTM [3, 37]	99.8	97.4	66.1	99.8	99.7	99.7
EntNet [38]	98.4	98.0	73.4	91.8	**100.0**	62.3
DNC [5]	99.4	83.8	69.5	**100.0**	**100.0**	**100.0**
Relational Memory Core	**99.9**	**99.6**	**79.0**	**100.0**	**100.0**	99.8

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

Results on LTE



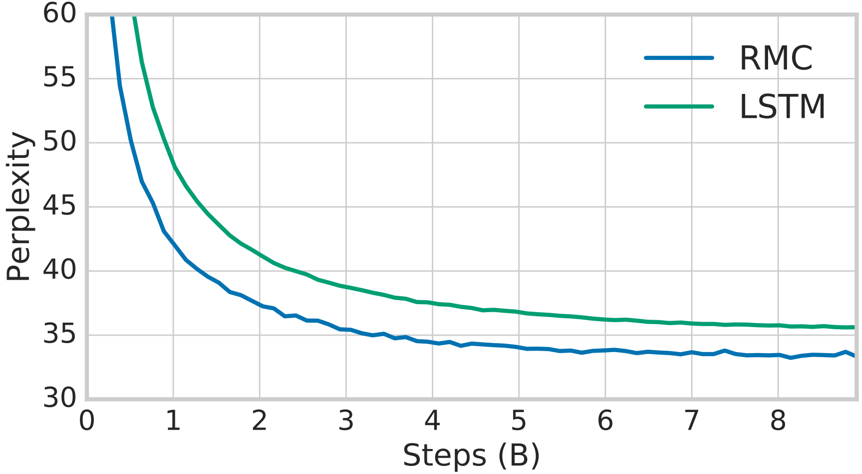
Results on Language Modelling

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

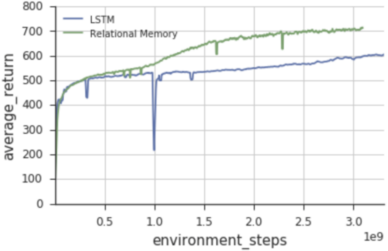
	WikiText-103		Gutenberg		GigaWord
	Valid.	Test	Valid	Test	Test
LSTM [40]	-	48.7	-	-	-
Temporal CNN [41]	-	45.2	-	-	-
Gated CNN [42]	-	37.2	-	-	-
LSTM [32]	34.1	34.3	41.8	45.5	43.7
Quasi-RNN [43]	32	33	-	-	-
Relational Memory Core	**30.8**	**31.6**	**39.2**	**42.0**	**38.3**

The RRNN obtains competitive perplexity levels, compared to several strong baselines.

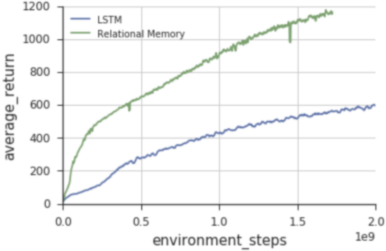
Results on Language Modelling: WikiText-103



Results on Mini-PacMan



With viewport



Without Viewport

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!

Concluding remarks

- 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.

Thank you!

Questions?

petar.velickovic@cst.cam.ac.uk

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