NEW YORK UNIVERSITY

Facebook AI Research

Deep Learning

Lecture 3 – Memory & Sets

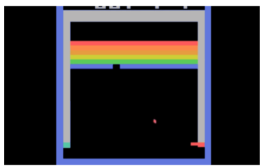
Machine Learning Summer School – Madrid 2018

Rob Fergus

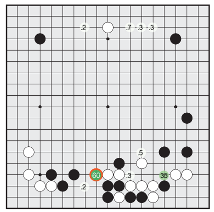
New York University
Facebook AI Research

Deep Learning Models

Spatial structure



[Mnih et al. 2015]



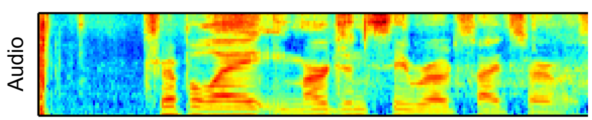
[Silver, Nature 16]

Convolutional Networks

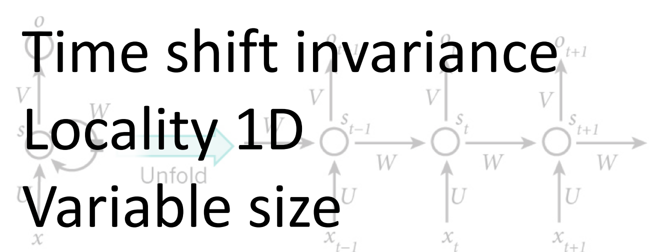
- Translation invariance
- Locality in 2D

Sequential structure

"Alexa, is it cold outside?"



Recurrent Networks



- Time shift invariance
- Locality 1D
- Variable size

[LeCun, Bengio & Hinton, Nature 2015]

<u>**Very specific structure to data**</u>

Overview

- Most effort focused on Deep Nets for perceptual problems
  - E.g. vision, speech, NLP.
- But what about other types of data?
  - Genome, 3D meshes, sets
- Or other aspects of intelligence?
  - Reasoning, e.g. program induction
  - Memory
  - Multi-agent learning / "Theory of mind"

Themes

- 1. Memory in Deep Nets

- 2. Deep Nets for sets

Memory Introduction

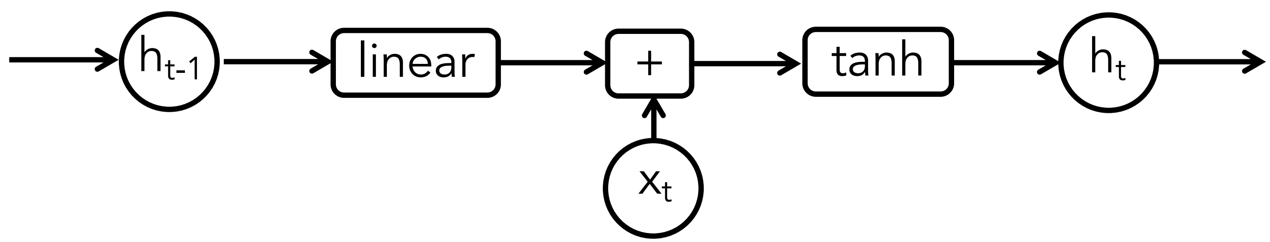
- Many tasks require some kind of memory
- But traditional neural networks are not good at remembering things, especially when input is large but only part of it is relevant
- Recently, there has been lot of interest in incorporating memory and attention to neural networks
  - Memory Networks, Neural Turing Machine,…

Memory Outline

- Implicit Internal memory
  - Recurrent Neural Nets (RNNs)
  - Long-Short Term Memory (LSTMs)
- Attention models
  - MT, Speech, Images
- Explicit External memory
  - Memory Networks
  - Neural Turing Machine
  - Stack-RNN
- Discrete Memory
  - 1-D tape, 2-D grid

Implicit Internal Memory

- Internal state of the model can be used for memory
  - Recurrent Neural Networks (RNNs)



- Computation and memory is mixed
  - Complex computation requires many layers of non-linearity
  - But some information is lost with each non-linearity
  - Problems with vanishing/exploding gradients & catastrophic forgetting

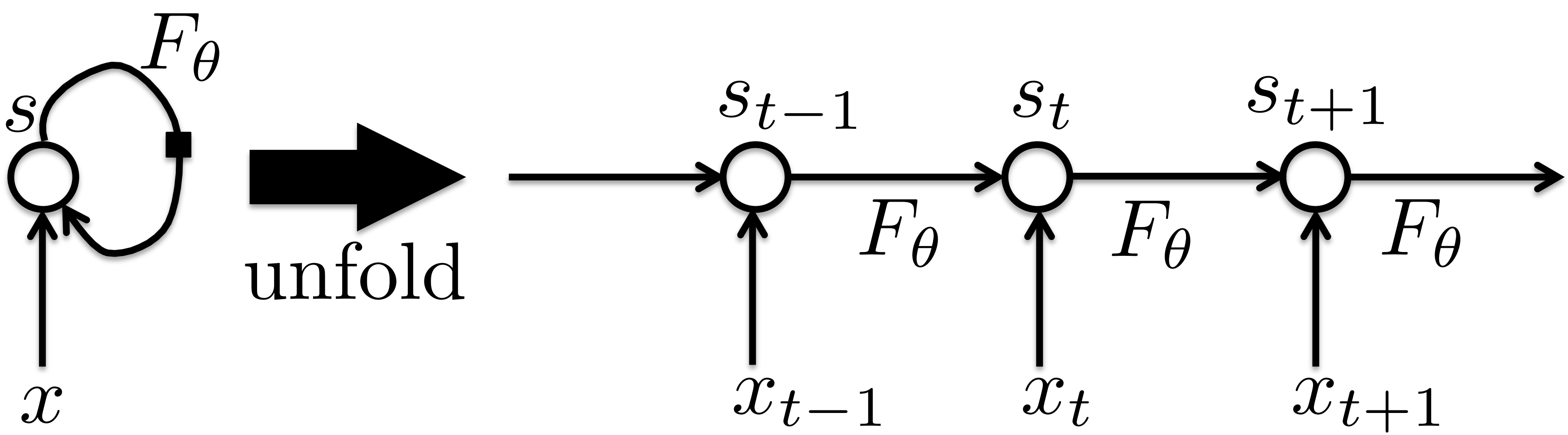
Memory Outline

- Implicit Internal memory
  - Recurrent Neural Nets (RNNs)
  - Long-Short Term Memory (LSTMs)
- Attention models
  - MT, Speech, Images
- Explicit External memory
  - Memory Networks
  - Neural Turing Machine
  - Stack-RNN
- Discrete Memory
  - 1-D tape, 2-D grid

Recurrent Neural Networks

- Selectively summarize an input sequence in a fixed-size state vector via a recursive update

$$s_t = F_\theta(s_{t-1}, x_t)$$

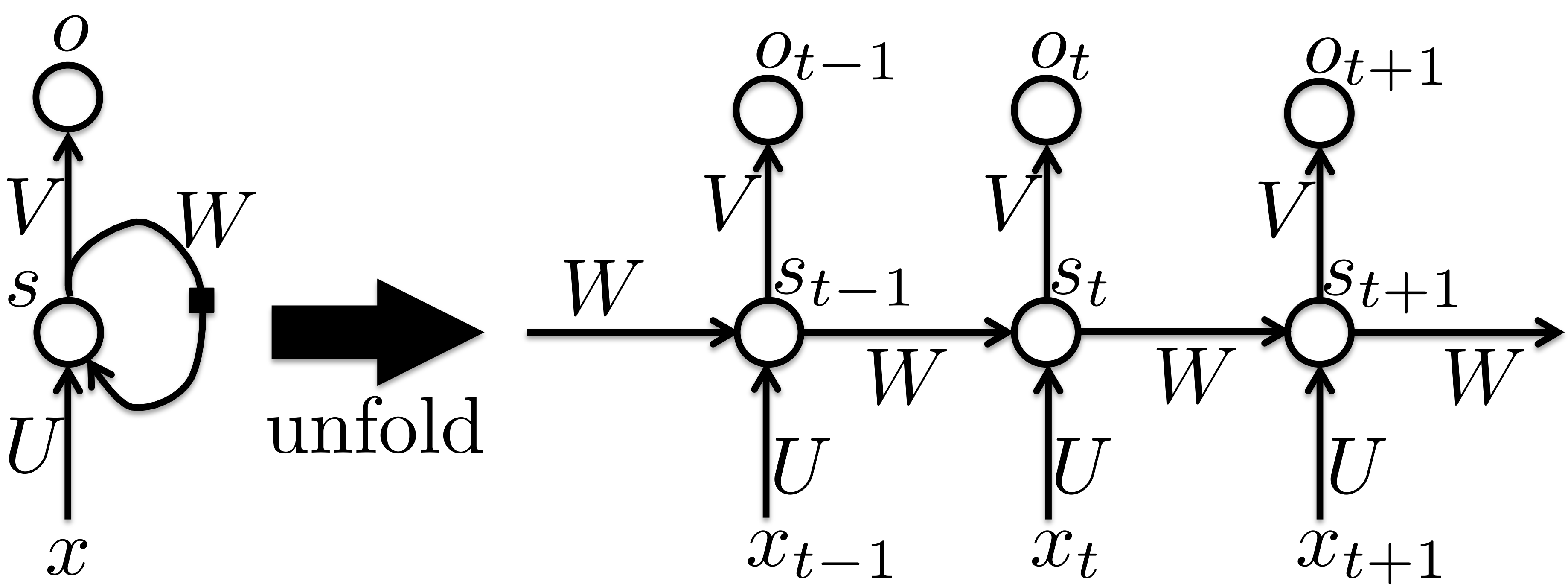


$$s_t = G_t(x_t, x_{t-1}, x_{t-2}, \ldots, x_2, x_1)$$

[Slide credit: Yoshua Bengio]

Recurrent Neural Networks

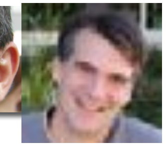
- Can produce an output at each time step: unfolding the graph tells us how to back-prop through time.



[Slide credit: Yoshua Bengio]

Long-Term Dependencies





- The RNN gradient is a product of Jacobian matrices, each associated with a step in the forward computation. To store information robustly in a finite-dimensional state, the dynamics must be contractive [Bengio et al 1994].

$$L = L(s_T(s_{T-1}(\ldots s_{t+1}(s_t, \ldots))))$$

$$\frac{\partial L}{\partial s_t} = \frac{\partial L}{\partial s_T} \frac{\partial s_T}{\partial s_{T-1}} \cdots \frac{\partial s_{t+1}}{\partial s_t}$$

Storing bits robustly requires sing. values<1

- Problems:
  - sing. values of Jacobians > 1 → *gradients explode*
  - or sing. values < 1 → *gradients shrink & vanish*
  - or random → variance grows exponentially

Gradient clipping

(Hochreiter 1991)

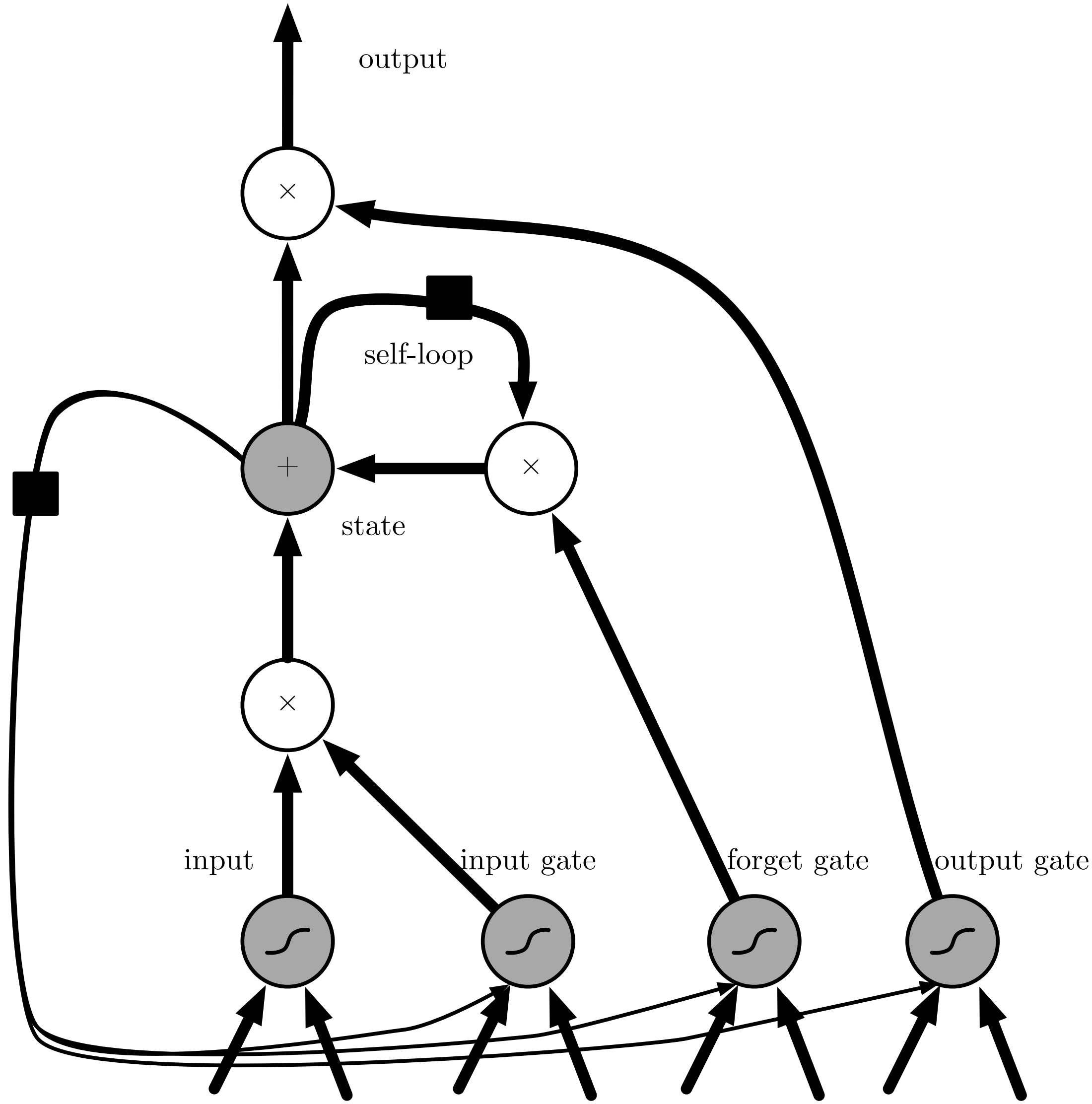
[Slide credit: Yoshua Bengio]

Ways to Prevent Forgetting in RNNs

- Split state into fast and slow changing parts: structurally constrained recurrent nets (e.g. Mikolov et al., 2014)
  - Fast changing part is good for computation
  - Slow changing part is good for storing information
- Gated units for internal state
  - Control when to forget/write using gates
  - Long-short term memory (LSTM) (see Graves, 2013)
  - Simpler Gated Recurrent Unit (GRU) (Cho et al., 2014)
- Other problems
  - Memory capacity is fixed and limited by the dimension of state vector (computation is $O(N^2)$ where N is memory capacity)
  - Vulnerable to distractions in inputs
  - Restricted to sequential inputs

Gated Recurrent Units & LSTM

- Create a path where gradients can flow for longer with self-loop
- Corresponds to an eigenvalue of Jacobian slightly less than 1
- LSTM is **heavily used** *(Hochreiter & Schmidhuber 1997)*
- GRU light-weight version *(Cho et al 2014)*



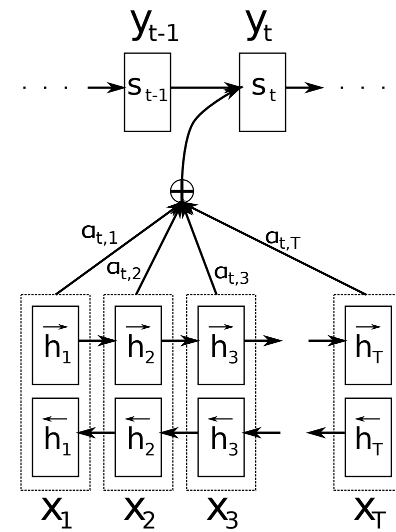
[Slide credit: Yoshua Bengio]

Memory Outline

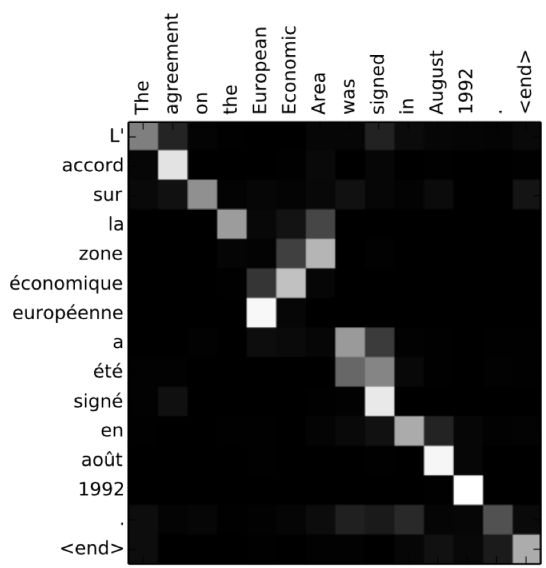
- Implicit Internal memory
  - Recurrent Neural Nets (RNNs)
  - Long-Short Term Memory (LSTMs)
- Attention models
  - MT, Speech, Images
- Explicit External memory
  - Memory Networks
  - Neural Turing Machine
  - Stack-RNN
- Discrete Memory
  - 1-D tape, 2-D grid

RNNsearch: Attention in Machine Translation (Bahdanau et al., 2015)

- RNN based encoder and decoder model
- Decoder can look at past encoder states using soft attention
- Attention mechanism is implement by a small neural network
  - It takes the current decoder state and a past encoder state and outputs a score. Then the all scores are fed to softmax to get attention weights
- Applied to machine translation. Significant improvement in translation of longer sentences



Attention weights during English to French machine translation



Significant improvement on long sentences

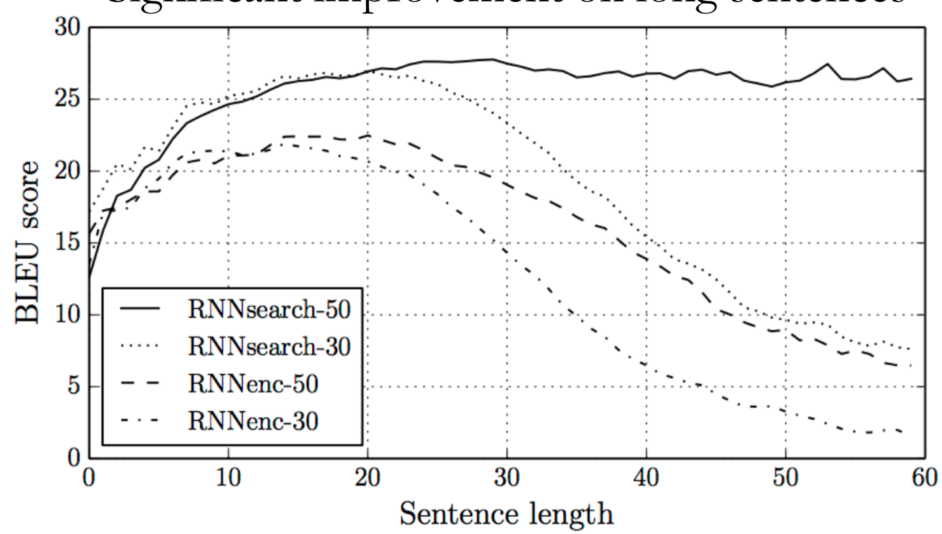
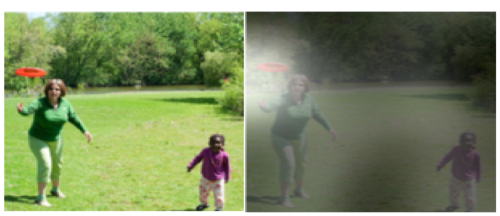
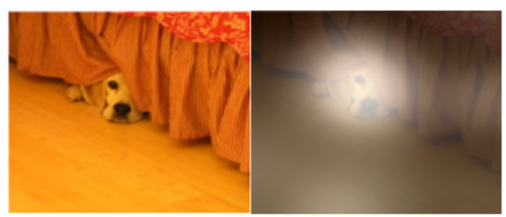


Image caption generation with attention (Xu et al., 2015)

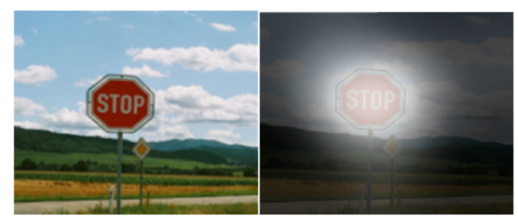
- Encoder: lower convolutional layer of a deep ConvNet (because need spatial information)
- Decoder: LSTM RNN with soft spatial attention
  - Decoder state and encoder state at single location are fed to small NN to get score at that location
- Network attends to the object when it is generating a word for it
- Also hard attention is tried with reinforcement learning



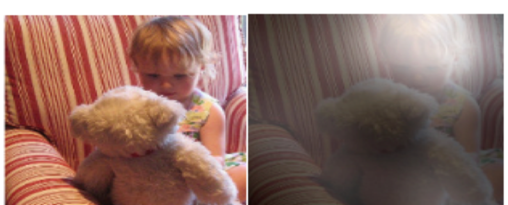
A woman is throwing a <u>frisbee</u> in a park.



A <u>dog</u> is standing on a hardwood floor.



A <u>stop</u> sign is on a road with a mountain in the background.

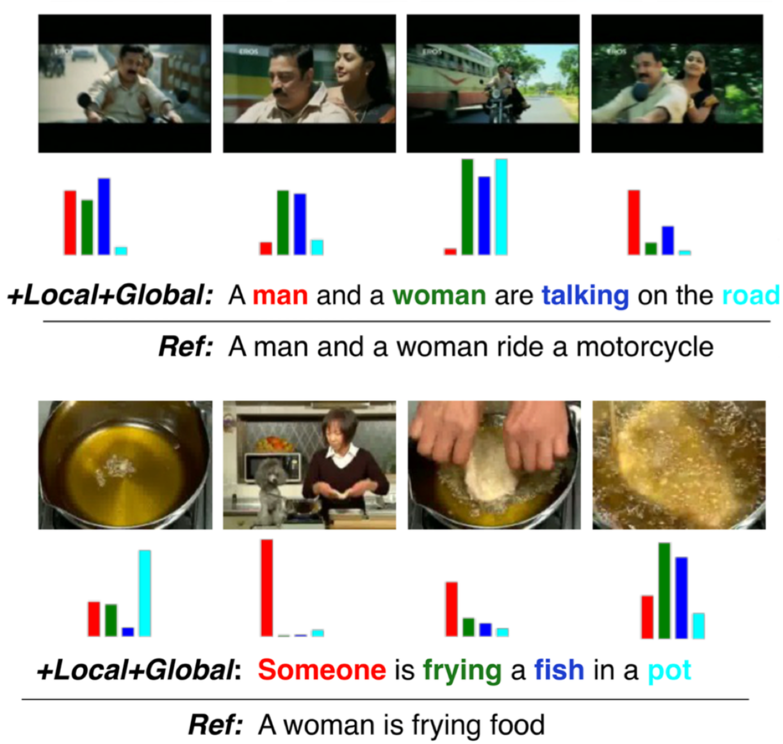


A little <u>girl</u> sitting on a bed with a teddy bear.

A group of <u>people</u> sitting on a boat in the water.

A giraffe standing in a forest with <u>trees</u> in the background.

Video description generation (Yao et al., 2015)

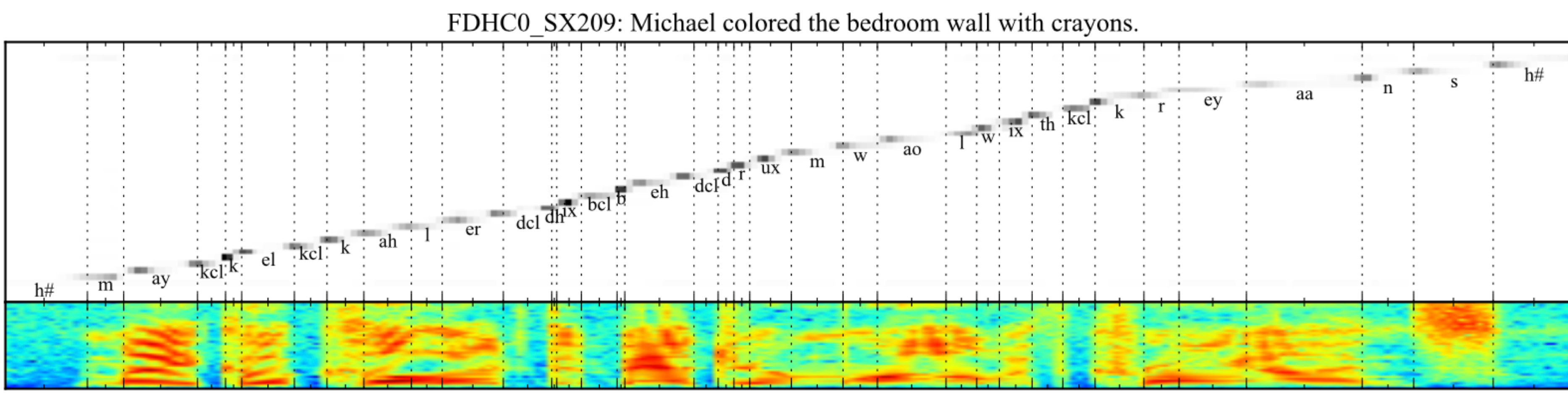


(bottom: ground truth)

L. Yao, A. Torabi, K. Cho, N. Ballas, C. Pal, H. Larochelle, and A. Courville, "Describing videos by exploiting temporal structure," *arXiv: 1502.08029*, 2015.

Location-aware attention for speech (Chorowski et al., 2015)

- RNN based encoder-decoder model with attention (similar to RNNsearch)
- Location based addressing: previous attention weights are used as feature for the current attention (good when subsequent attention locations are highly correlated)
- Improvement with sharpening and smoothing of memory addressing

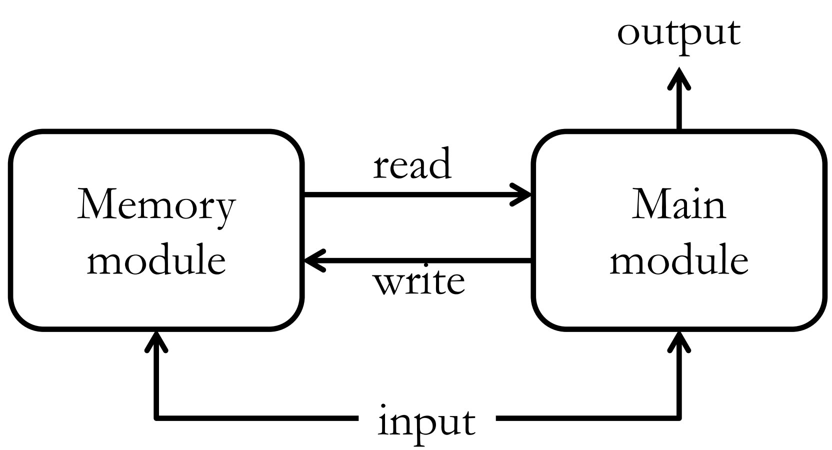


Outline

- Implicit Internal memory
  - Recurrent Neural Nets (RNNs)
  - Long-Short Term Memory (LSTMs)
- Explicit External memory
  - StackRNN
  - Memory Networks
  - Neural Turing Machine
- Attention models
  - MT, Speech, Image, Pointer Network
- Discrete Memory
  - Learning algorithms using 1-D tape, 2-D grid

External Global Memory

- Separate memory from computation
  - Add separate memory module for storage
  - Memory contains list/set of items



- Main module can read and write to the memory
- Advantage: long-term, scalable, flexible

Selective Addressing is Key for Memory

- Often, you only want to interact with few items in memory at once
  - Memory needs some addressing mechanism
- Memory addressing types
  - Soft or hard addressing
    - Soft addressing can be trained by backpropagation
    - Hard addressing is not differentiable (e.g. has to be trained with reinforcement learning or additional training signal for where to attend)
  - Context and Location based addressing
    - When input is ordered in some way, location based addressing is useful
    - Location addressing is same as context if location is embedded in the context (e.g. MemN2N)

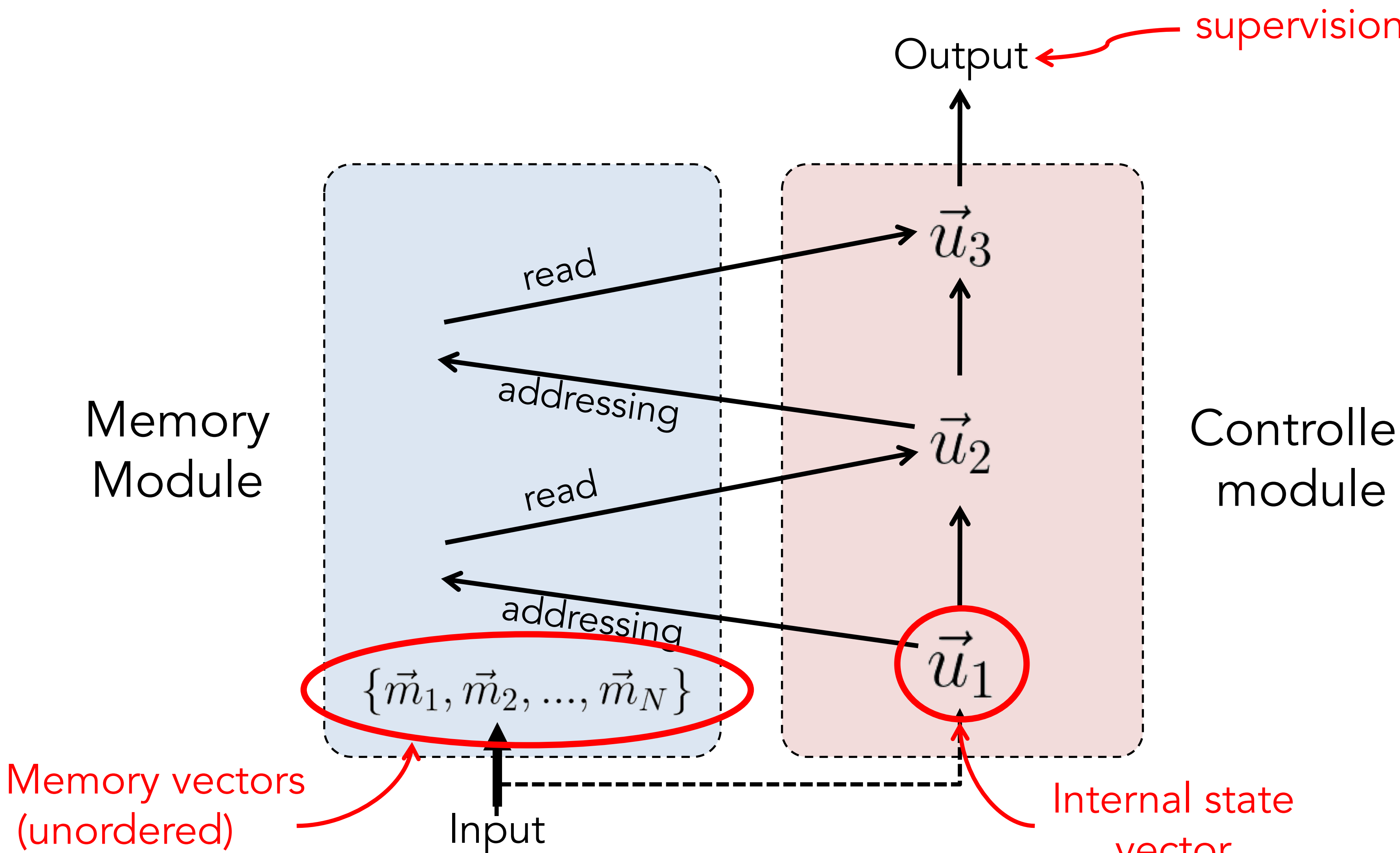
Memory Outline

- Implicit Internal memory
  - Recurrent Neural Nets (RNNs)
  - Long-Short Term Memory (LSTMs)
- Attention models
  - MT, Speech, Images
- Explicit External memory
  - Memory Networks
  - Neural Turing Machine
  - Stack-RNN
- Discrete Memory
  - 1-D tape, 2-D grid

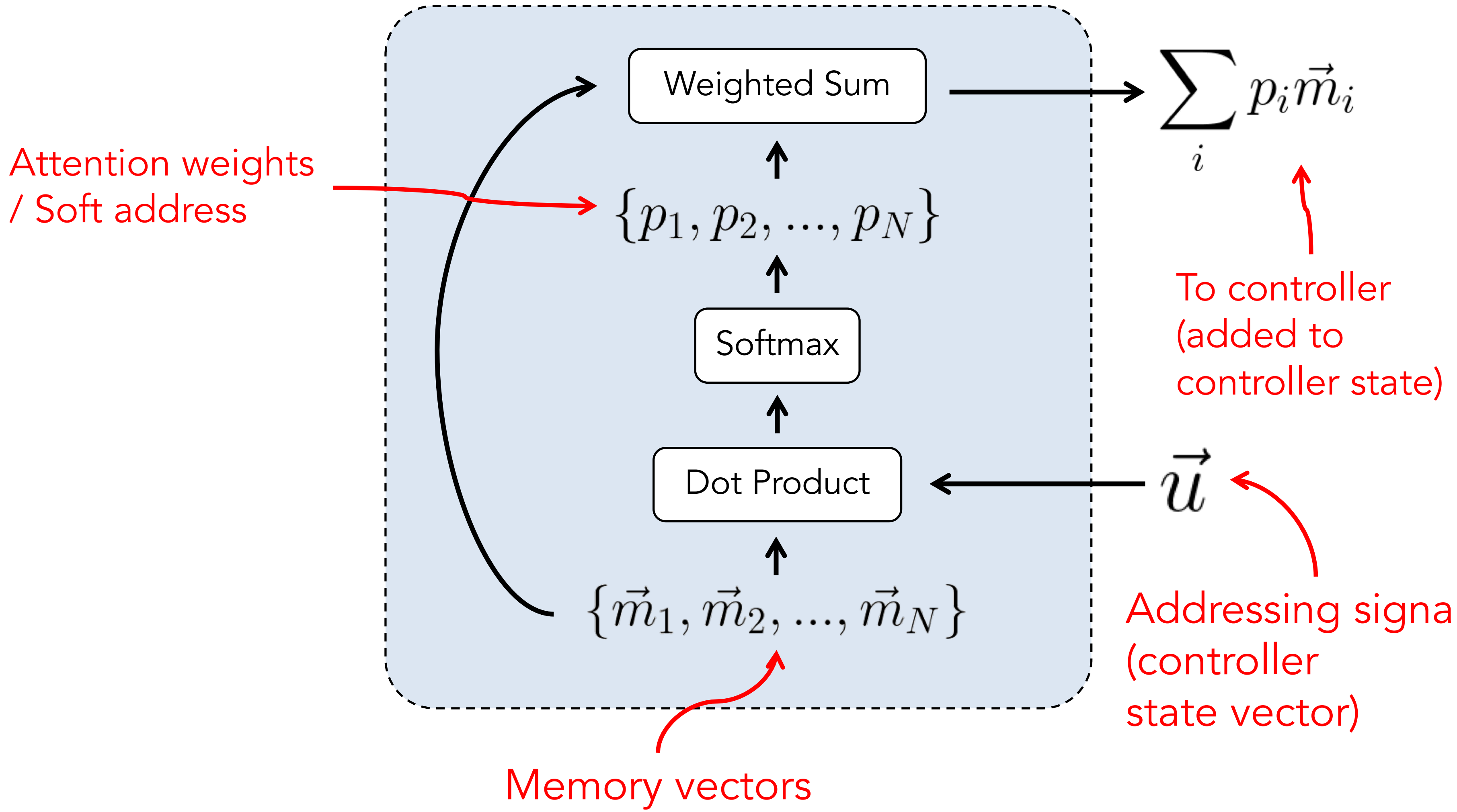
Memory Networks

- “Hard” Memory Networks by [Weston, Chopra & Bordes ICLR 2015]
  - Hard attention thus requires explicit supervision of attention during training
- End-to-end Memory Networks (MemN2N) has **soft** attention
  - Only need supervision on the final output
  - [Sukhbaatar et al., NIPS 2015]

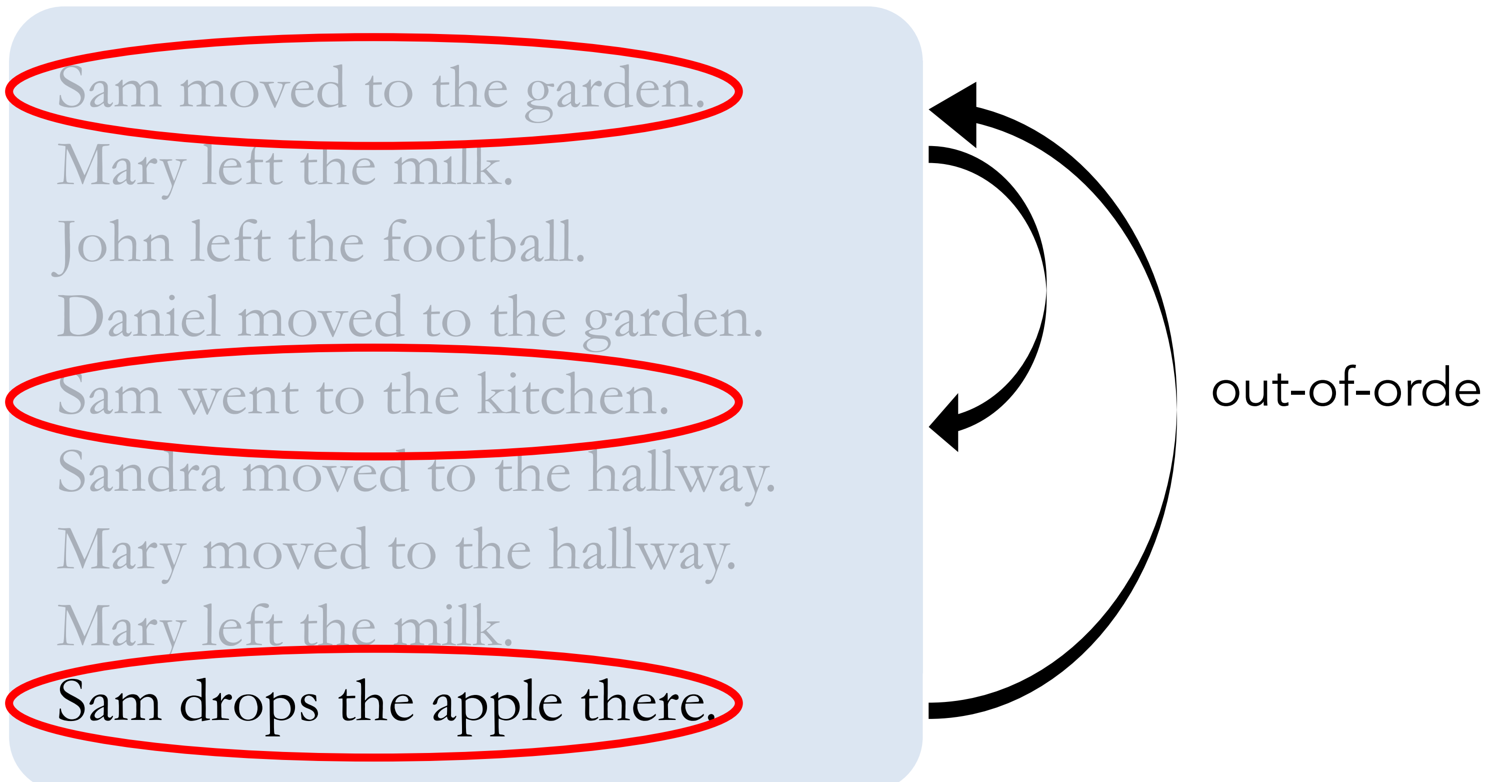
MemN2N architecture



Memory Module



Ex) Question & Answering on story



Q: Where was the apple after the garden?

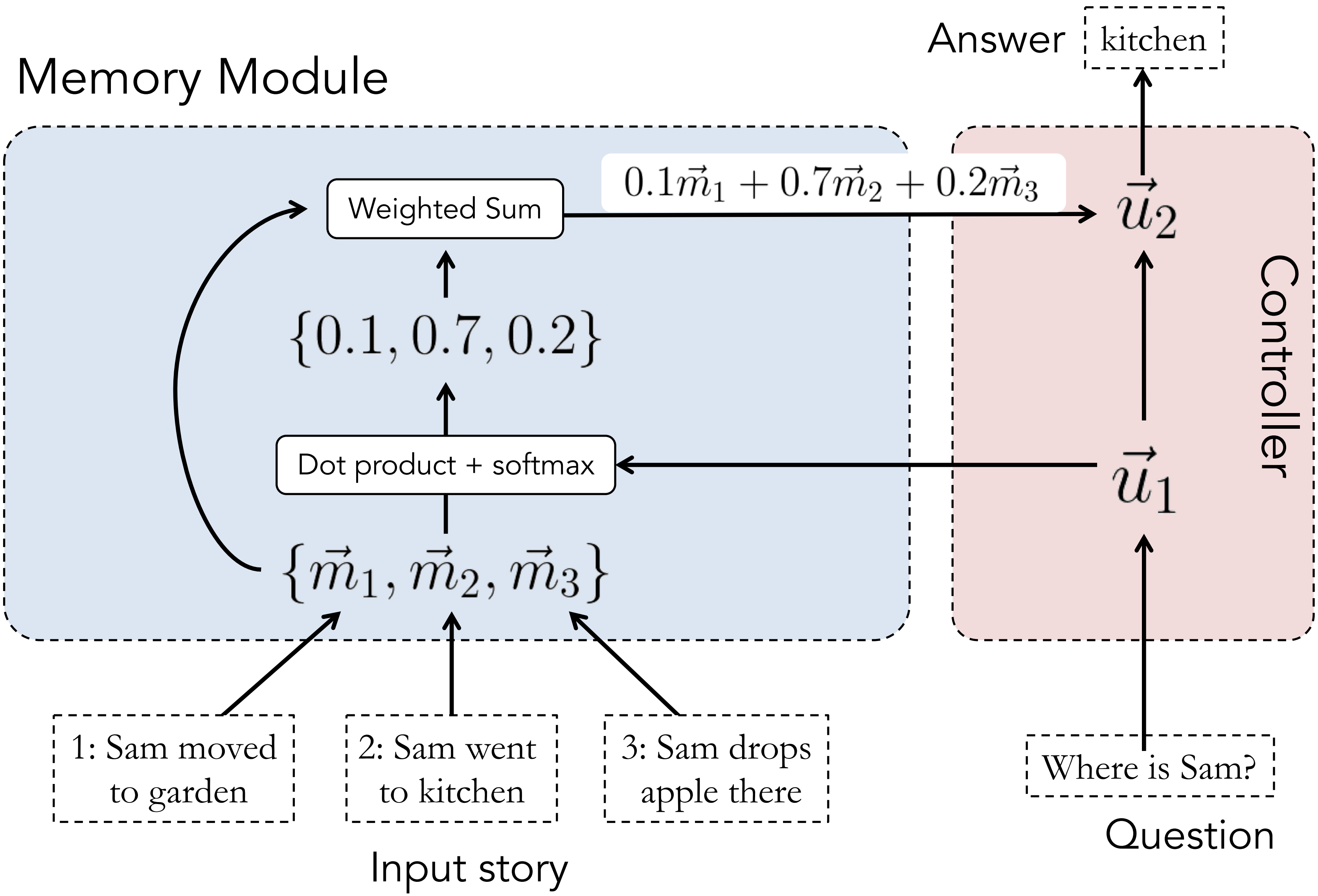
Memory Vectors

E.g.) constructing memory vectors with Bag-of-Words (BoW)

1. Embed each word
2. Sum embedding vectors

“Sam drops apple”

Question & Answering



Related Work

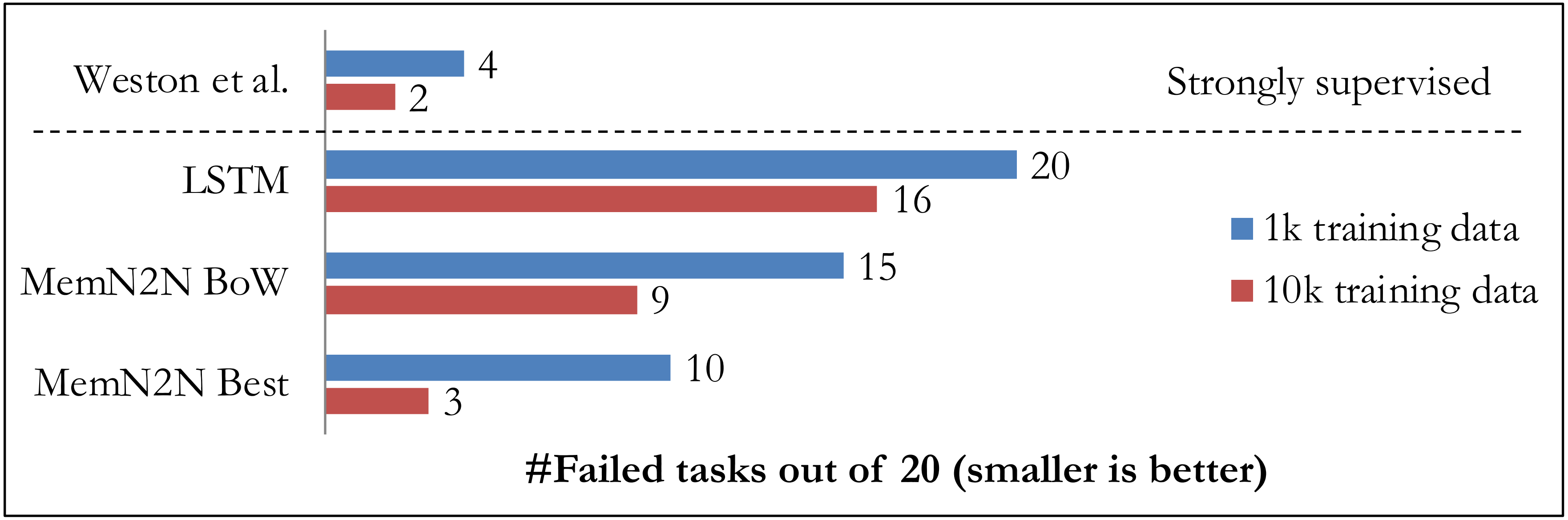
- RNNsearch [Bahdanau et al. 2015]
  - Encoder-decoder RNN with attention
  - Our model can be considered as an attention model with multiple hops
- Recent works on external memory
  - Stack memory for RNNs [Joulin & Mikolov. 2015]
  - Neural Turing Machine [Graves et al. 2014]
- Early works on neural network and memory
  - [Steinbuch & Piske. 1963]; [Taylor. 1959]
  - [Das et al. 1992]; [Mozer et al. 1993]
- Concurrent works
  - Dynamic Memory Networks [Kumar et al. 2015]
  - Attentive reader [Hermann et al. 2015]
  - Stack, Queue [Grefenstette et al. 2015]

Experiment on bAbI Q&A data

- Data: 20 bAbI tasks [Weston et al. arXiv: 1502.05698, 2015]
- Answer questions after reading short story
- Small vocabulary, simple language
- Different tasks require different reasoning
- Training data size 1K or 10K for each task

Sam walks into the kitchen. Sam picks up an apple. Sam walks into the bedroom. Sam drops the apple. Q: Where is the apple? A. Bedroom	Brian is a lion. Julius is a lion. Julius is white. Bernhard is green. Q: What color is Brian? A. White

Performance on bAbI test set



Examples of Attention Weights

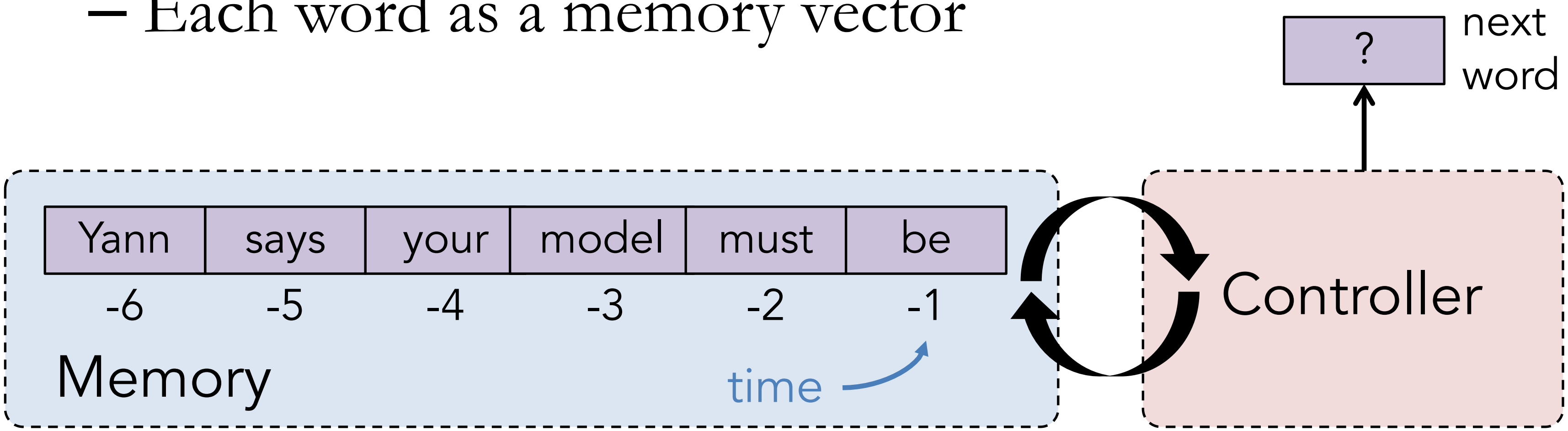
- 2 test cases:

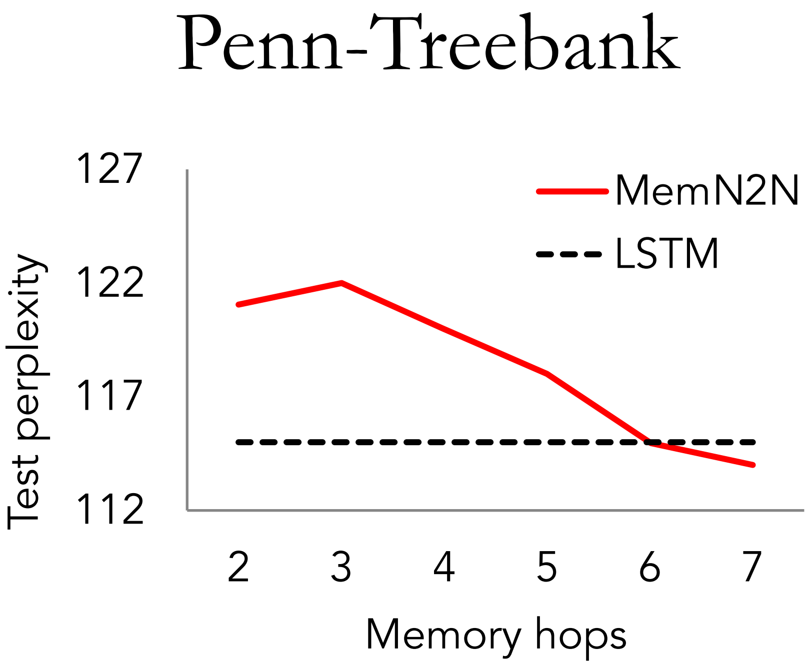
Story (2: 2 supporting facts)	**Hop 1**	**Hop 2**	**Hop 3**
John dropped the milk.	0.06	0.00	0.00
John took the milk there.	0.88	1.00	0.00
Sandra went back to the bathroom.	0.00	0.00	0.00
John moved to the hallway.	0.00	0.00	1.00
Mary went back to the bedroom.	0.00	0.00	0.00
Where is the milk? Answer: hallway Prediction: hallway			

Story (16: basic induction)	**Hop 1**	**Hop 2**	**Hop 3**
Brian is a frog.	0.00	0.98	0.00
Lily is gray.	0.07	0.00	0.00
Brian is yellow.	0.07	0.00	1.00
Julius is green.	0.06	0.00	0.00
Greg is a frog.	0.76	0.02	0.00
What color is Greg? Answer: yellow Prediction: yellow			

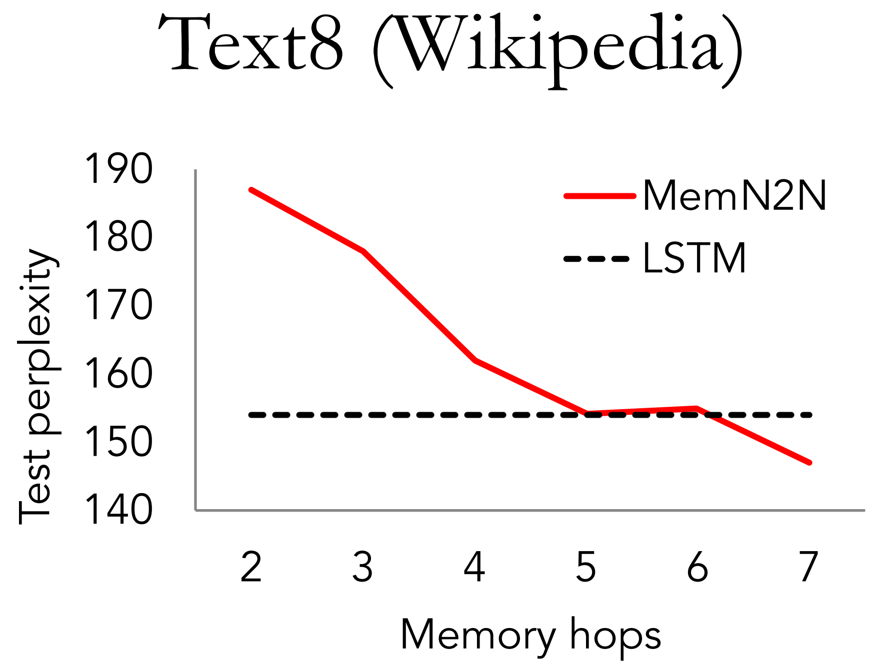
Experiment on Language modeling

- Data
  - Penn Treebank: 1M words 10K vocab
  - Text8 (Wikipedia): 16M words 40K vocab
- Model
  - Controller module: linear + non-linearity
  - Each word as a memory vector



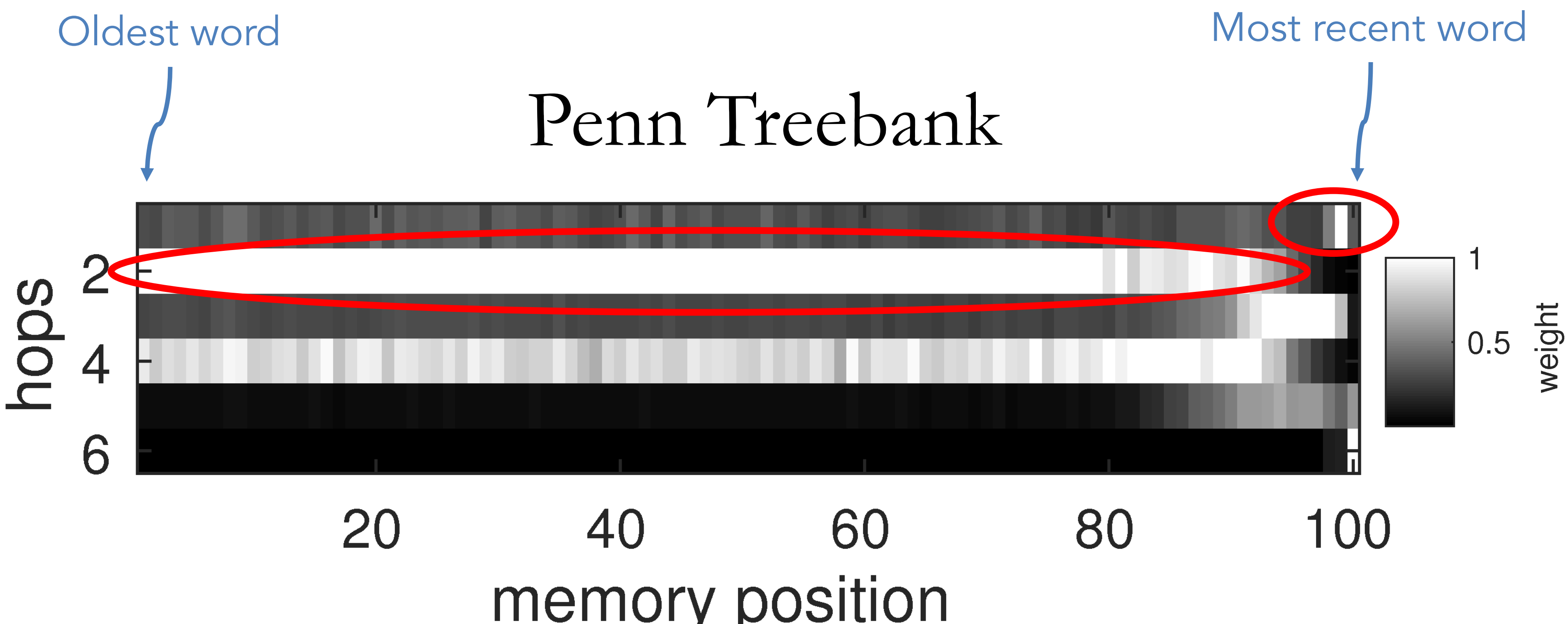


Penn-Treebank
127
122
117
112
Test perplexity
MemN2N
LSTM
2
3
4
5
6
7
Memory hops



Text8 (Wikipedia)
190
180
170
160
150
140
Test perplexity
MemN2N
LSTM
2
3
4
5
6
7
Memory hops

Attention during memory hops

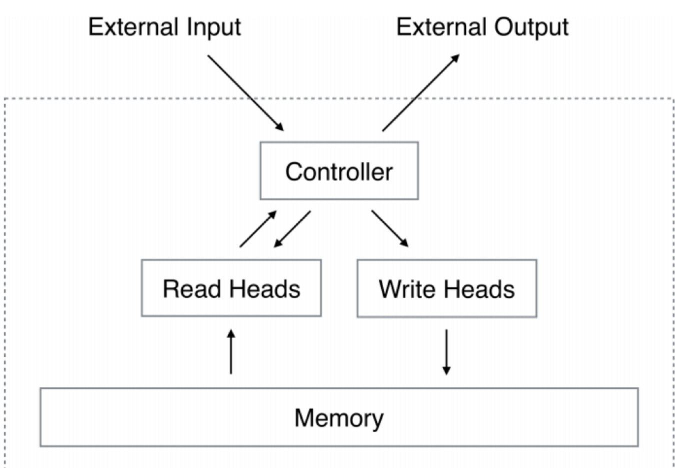


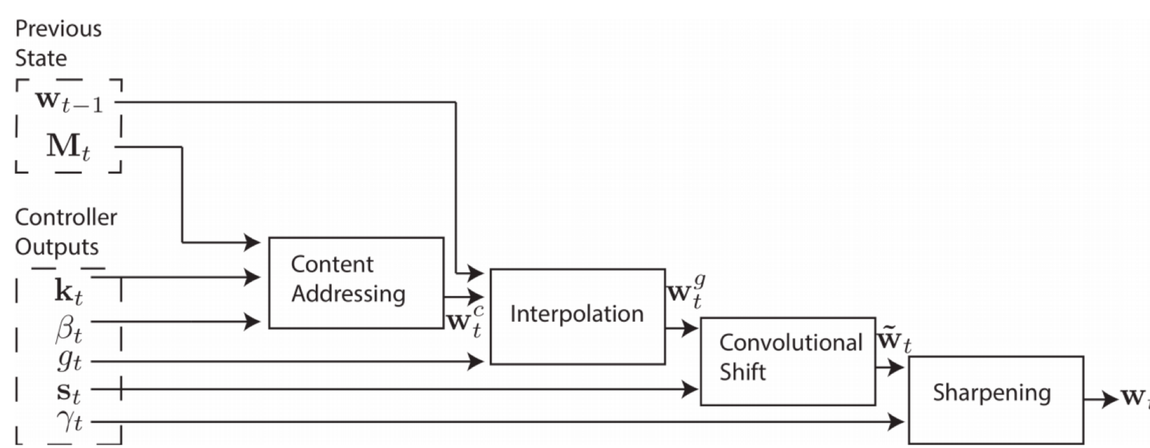
Memory Outline

- Implicit Internal memory
  - Recurrent Neural Nets (RNNs)
  - Long-Short Term Memory (LSTMs)
- Attention models
  - MT, Speech, Images
- Explicit External memory
  - Memory Networks
  - Neural Turing Machine
  - Stack-RNN
- Discrete Memory
  - 1-D tape, 2-D grid

Neural Turing Machine (Graves et al., 2014)

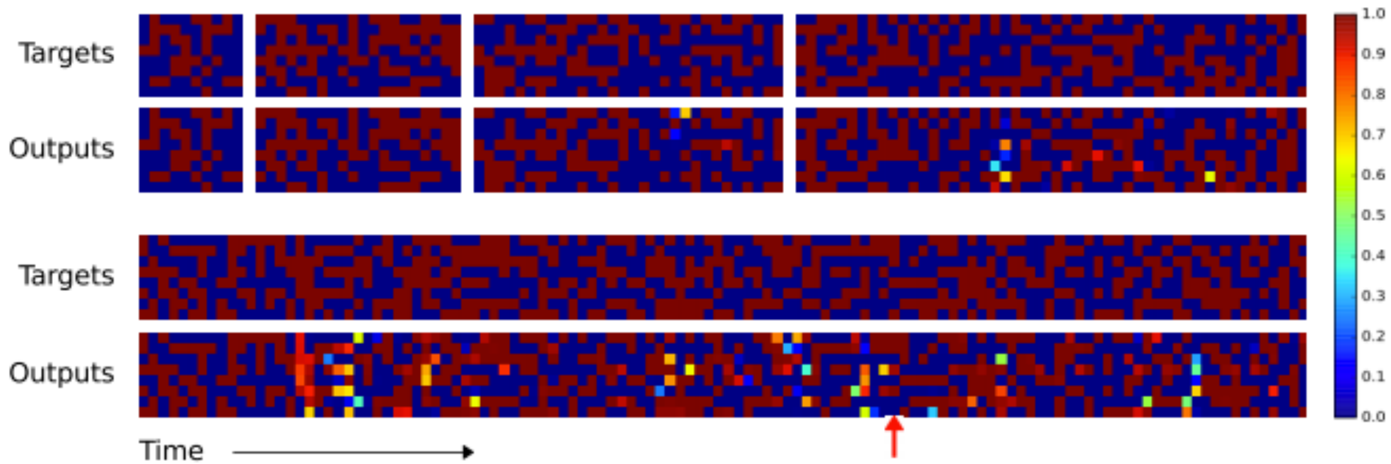
- Learns how to write to the memory
- Soft addressing → backpropagation training
- Location addressing: small continuous shift of attention
- Complex addressing mechanism: need to sharpen after convolution
- Controller can be LSTM-RNN or feed-forward neural network
- Applied to learn algorithms such as sort, associative recall and copy.
- Also hard addressing with reinforcement learning [Zaremba et al., 2015]
- Also Differentiable Neural Computer [Graves et al., 2016]

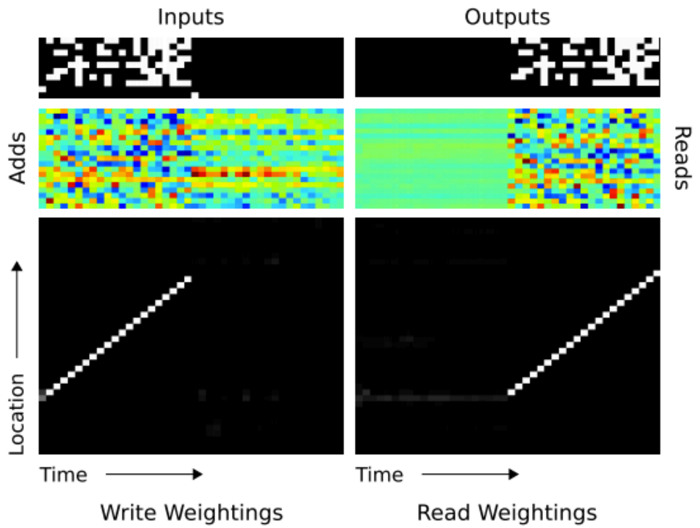




Neural Turing Machine – Copy task

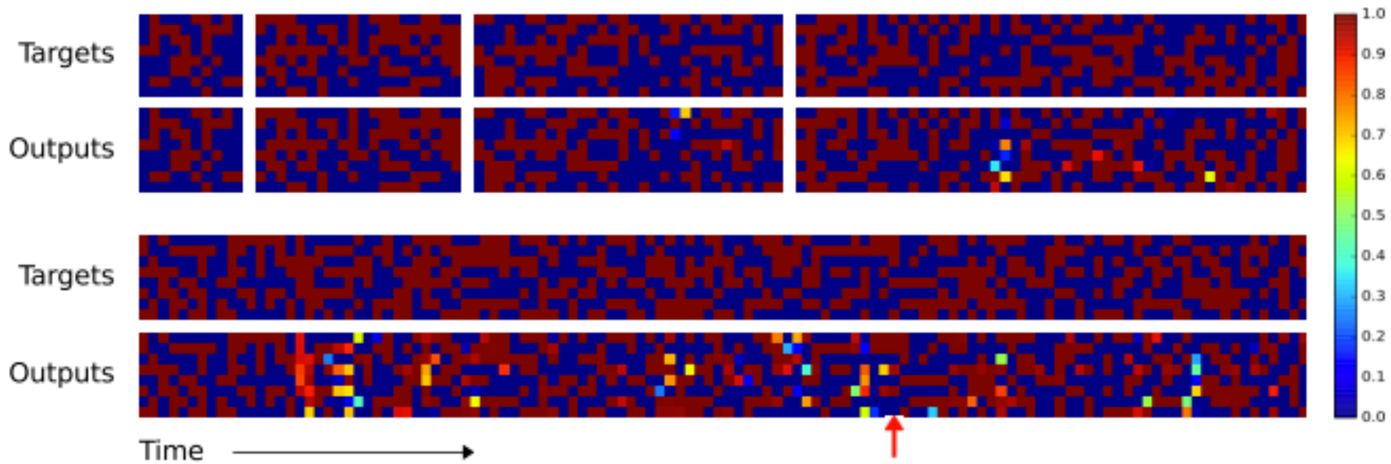
- NTM



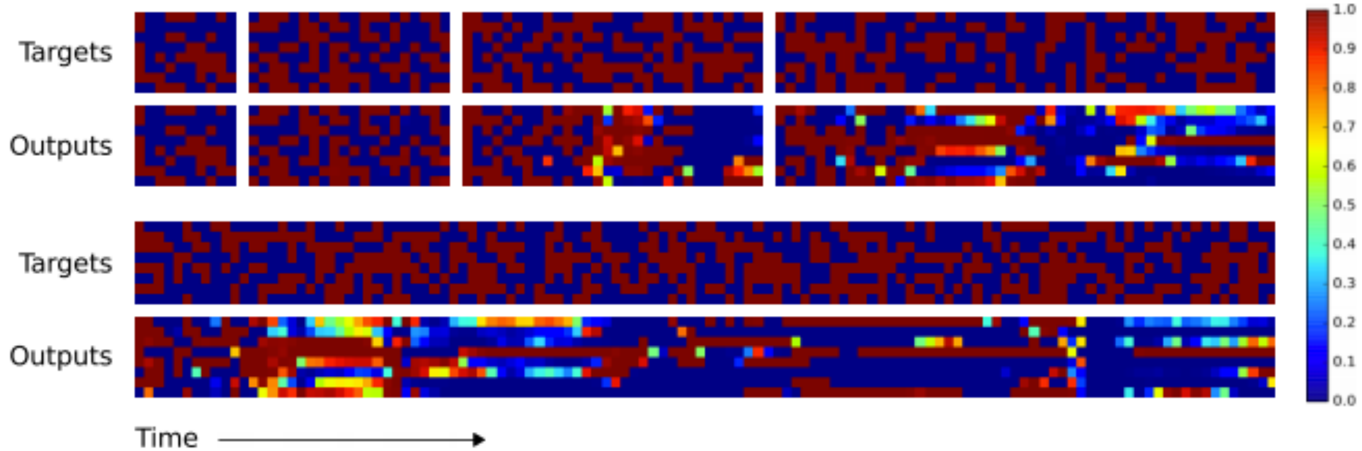


Neural Turing Machine – Copy task

- NTM



LSTM



Neural Turing Machine - Experiments

Task	#Heads	Controller Size	Memory Size	Learning Rate	#Parameters
Copy	1	100	128×20	10^{-4}	$17,162$
Repeat Copy	1	100	128×20	10^{-4}	$16,712$
Associative	4	256	128×20	10^{-4}	$146,845$
N-Grams	1	100	128×20	3×10^{-5}	$14,656$
Priority Sort	8	512	128×20	3×10^{-5}	$508,305$

Table 1: NTM with Feedforward Controller Experimental Settings

Memory Outline

- Implicit Internal memory
  - Recurrent Neural Nets (RNNs)
  - Long-Short Term Memory (LSTMs)
- Attention models
  - MT, Speech, Images
- Explicit External memory
  - Memory Networks
  - Neural Turing Machine
  - Stack-RNN
- Discrete Memory
  - 1-D tape, 2-D grid

Stack RNNs (Joulin & Mikolov, 2015)

- Simple RNN extended with a stack that the neural net learns to control

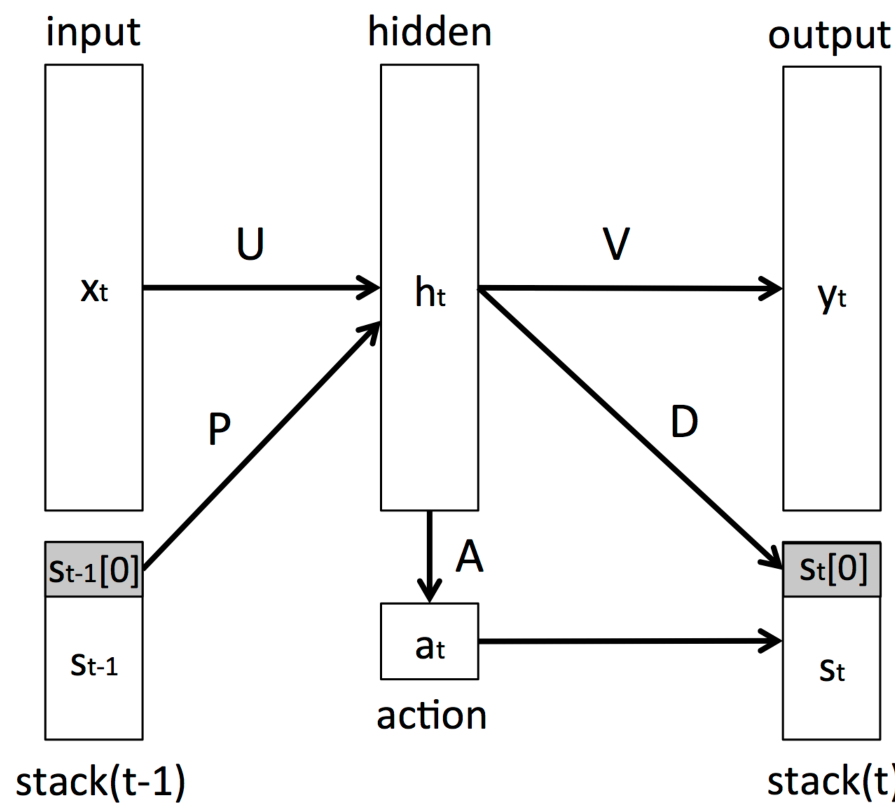
- The idea itself is very old (from 80's – 90's)

- Very simple and learns complex toy patterns with much less supervision & scales to more complex tasks

Stack RNN

- Add structured memory to RNN:
  - Trainable [read/write]
  - Unbounded
- Continuous actions:
  PUSH / POP / NO-OP
- Multiple stacks

- Examples of memory structures:
  stacks, lists, queues, tapes, grids, …
- Learns algorithms from examples



Stack RNN - Algorithmic Patterns

Sequence generator	Example
$\{a^n b^n \mid n > 0\}$	aab**ba**aab**bba**b**a**aaaab**bbbb**
$\{a^n b^n c^n \mid n > 0\}$	aaab**bbccca**b**ca**aaaab**bbbbccccc**
$\{a^n b^n c^n d^n \mid n > 0\}$	aab**bccdda**aab**bbcccddda**b**cd**
$\{a^n b^{2n} \mid n > 0\}$	aab**bbba**aab**bbbbba**b**b**
$\{a^n b^m c^{n+m} \mid n, m > 0\}$	aabc**cca**aabbc**cccca**bc**c**
$n \in [1, k],\ X \to nXn,\ X \to =$	$(k = 2)$ 12=**21**2122=**2212**11121=**12111**

- Examples of simple algorithmic patterns generated by short programs (grammars)

- The goal is to learn these patterns in an **unsupervised** manner just by observing the example sequences

Stack RNN - Example

- Sequence: a^6b^{12}

current	next	prediction	proba(next)	action		stack1[top]	stack2[top]
b	a	a	0.99	POP	POP	-1	0.53
a	a	a	0.99	PUSH	POP	0.01	0.97
a	a	a	0.95	PUSH	PUSH	0.18	0.99
a	a	a	0.93	PUSH	PUSH	0.32	0.98
a	a	a	0.91	PUSH	PUSH	0.40	0.97
a	a	a	0.90	PUSH	PUSH	0.46	0.97
a	b	a	0.10	PUSH	PUSH	0.52	0.97
b	**b**	**b**	0.99	PUSH	PUSH	0.57	0.97
b	**b**	**b**	1.00	POP	PUSH	0.52	0.56
b	**b**	**b**	1.00	POP	PUSH	0.46	0.01
b	**b**	**b**	1.00	POP	PUSH	0.40	0.00
b	**b**	**b**	1.00	POP	PUSH	0.32	0.00
b	**b**	**b**	1.00	POP	PUSH	0.18	0.00
b	**b**	**b**	0.99	POP	PUSH	0.01	0.00
b	**b**	**b**	0.99	POP	POP	-1	0.00
b	**b**	**b**	0.99	POP	POP	-1	0.00
b	**b**	**b**	0.99	POP	POP	-1	0.00
b	**b**	**b**	0.99	POP	POP	-1	0.01
b	**a**	**a**	0.99	POP	POP	-1	0.56

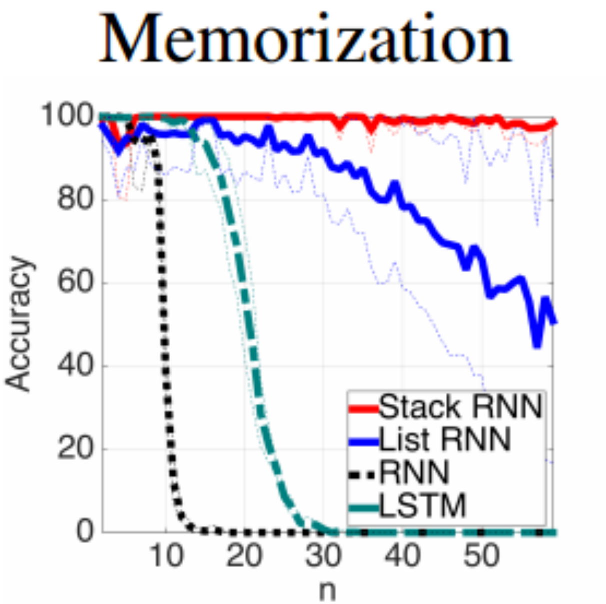
Table 3: Example of the Stack RNN with 20 hidden units and 2 stacks on a sequence a^nb^{2n} with $n = 6$. -1 means that the stack is empty. The depth k is set to 1 for clarity. We see that the first stack pushes an element every time it sees a and pop when it sees b. The second stack pushes when it sees a. When it sees b , it pushes if the first stack is not empty and pop otherwise. This shows how the two stacks interact to correctly predict the deterministic part of the sequence (shown in bold).

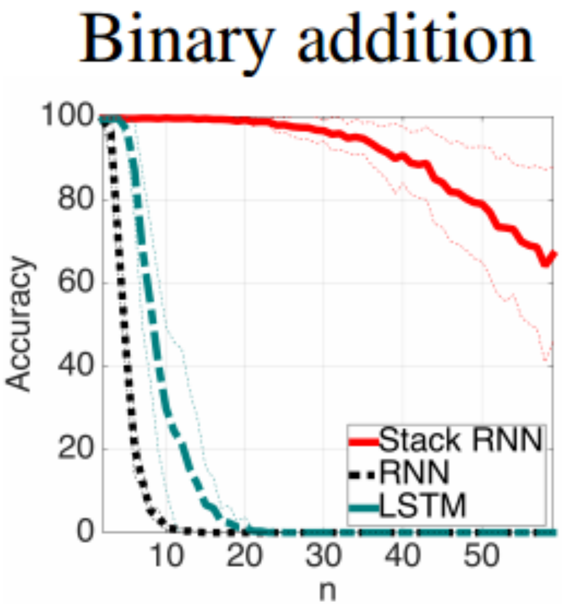
Algorithmic Patterns - Counting

method	$a^n b^n$	$a^n b^n c^n$	$a^n b^n c^n d^n$	$a^n b^{2n}$	$a^n b^m c^{n+m}$
RNN	25%	23.3%	13.3%	23.3%	33.3%
LSTM	100%	100%	68.3%	75%	100%
List RNN 40+5	100%	33.3%	100%	100%	100%
Stack RNN 40+10	100%	100%	100%	100%	43.3%
Stack RNN 40+10 + rounding	100%	100%	100%	100%	100%

- Performance on simple counting tasks
- RNN with sigmoidal activation function cannot count
- Stack-RNN and LSTM can count

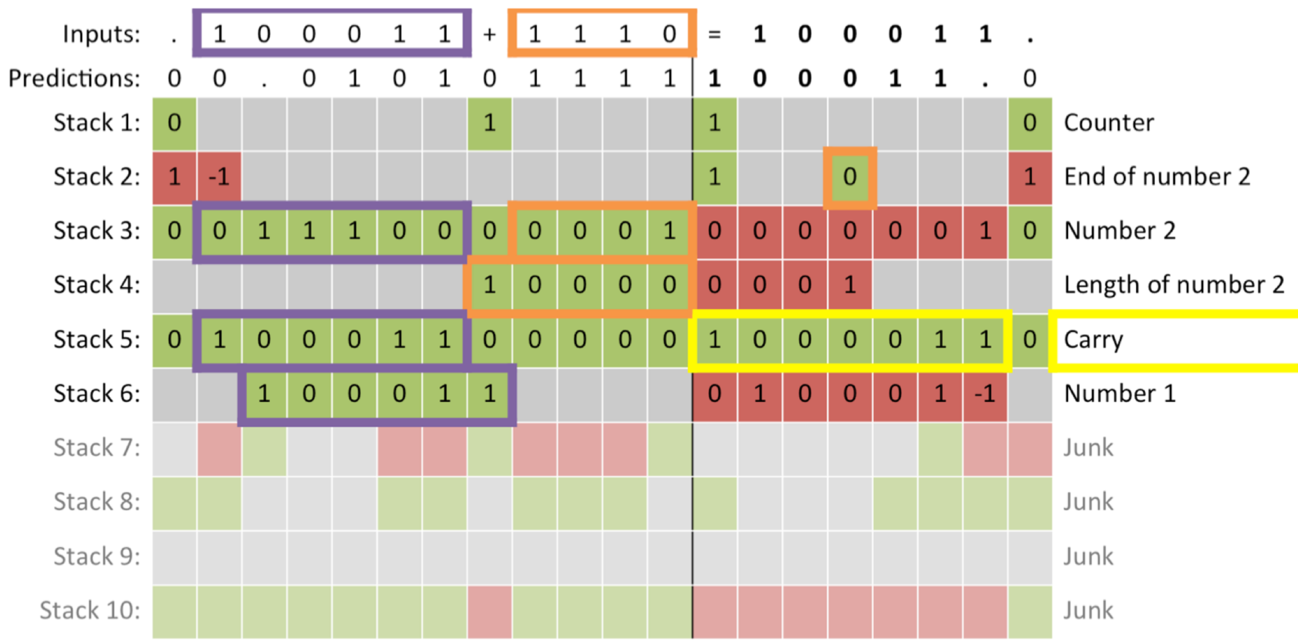
Algorithmic Patterns - Sequences





- Sequence memorization and binary addition are out-of-scope of LSTM
- Expandable memory of stacks allows to learn the solution

Stack RNN - Binary Addition



- **No supervision in training, just prediction**
- Learns to: store digits, when to produce output, carry

Stack RNNs: summary

The good:

- Turing-complete model of computation (with >=2 stacks)
- Learns some algorithmic patterns
- Has long term memory
- Works for some problems that break RNNs and LSTMs
- Reproducible: https://github.com/facebook/Stack-RNN

The bad:

- The long term memory is used only to store partial computation (ie. learned skills are not stored there yet)
- Does not seem to be a good model for incremental learning due to computational inefficiency of the model
- Stacks do not seem to be a very general choice for the topology of the memory

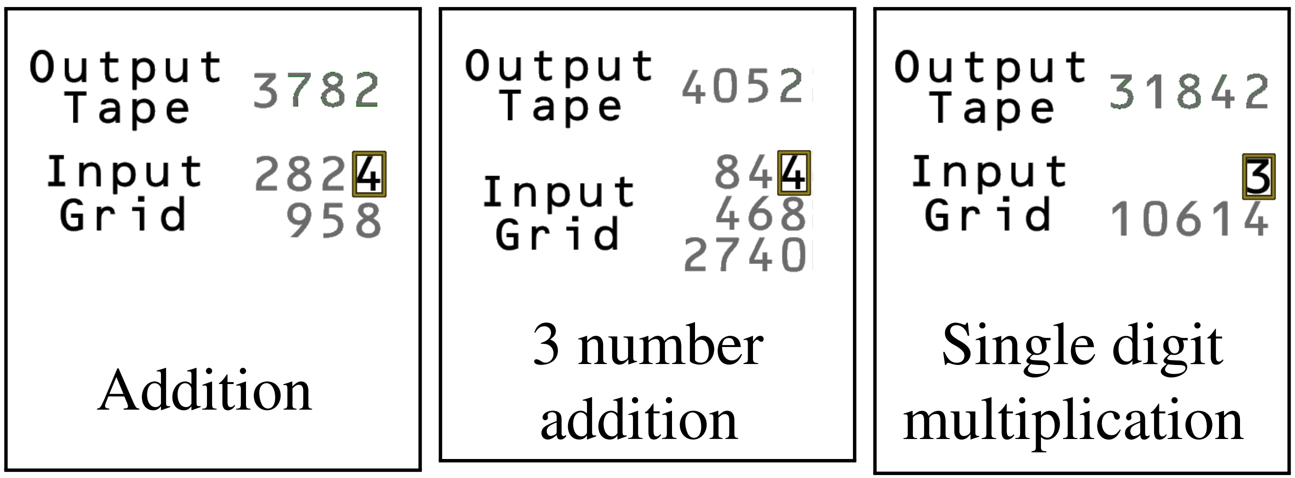
Memory Outline

- Implicit Internal memory
  - Recurrent Neural Nets (RNNs)
  - Long-Short Term Memory (LSTMs)
- Attention models
  - MT, Speech, Images
- Explicit External memory
  - Memory Networks
  - Neural Turing Machine
  - Stack-RNN
- Discrete Memory
  - 1-D tape, 2-D grid

Learning Simple Algorithms from Examples

[Zaremba et al. ICML 2016]

- Given examples of simple addition, multiplication etc can we learn the underlying algorithm?
  - Must generalize to much longer examples



Model Setup

- Explore various controllers (1 layer, 200 units)
  - Feed forward, LSTM, GRU
  - Additional linear layer predicts symbol
- Choose interfaces appropriate for task
- Dual output from controller:

1. Discrete actions (“move output head left”, “do nothing”)
   - Trained using reinforcement learning
   - Don’t get label until output a digit

2. Continuous prediction of symbol
   - Backpropagation through softmax output

Solving the tasks with Reinforcement Learning

- Sequence of discrete actions taken to produce symbol at output.

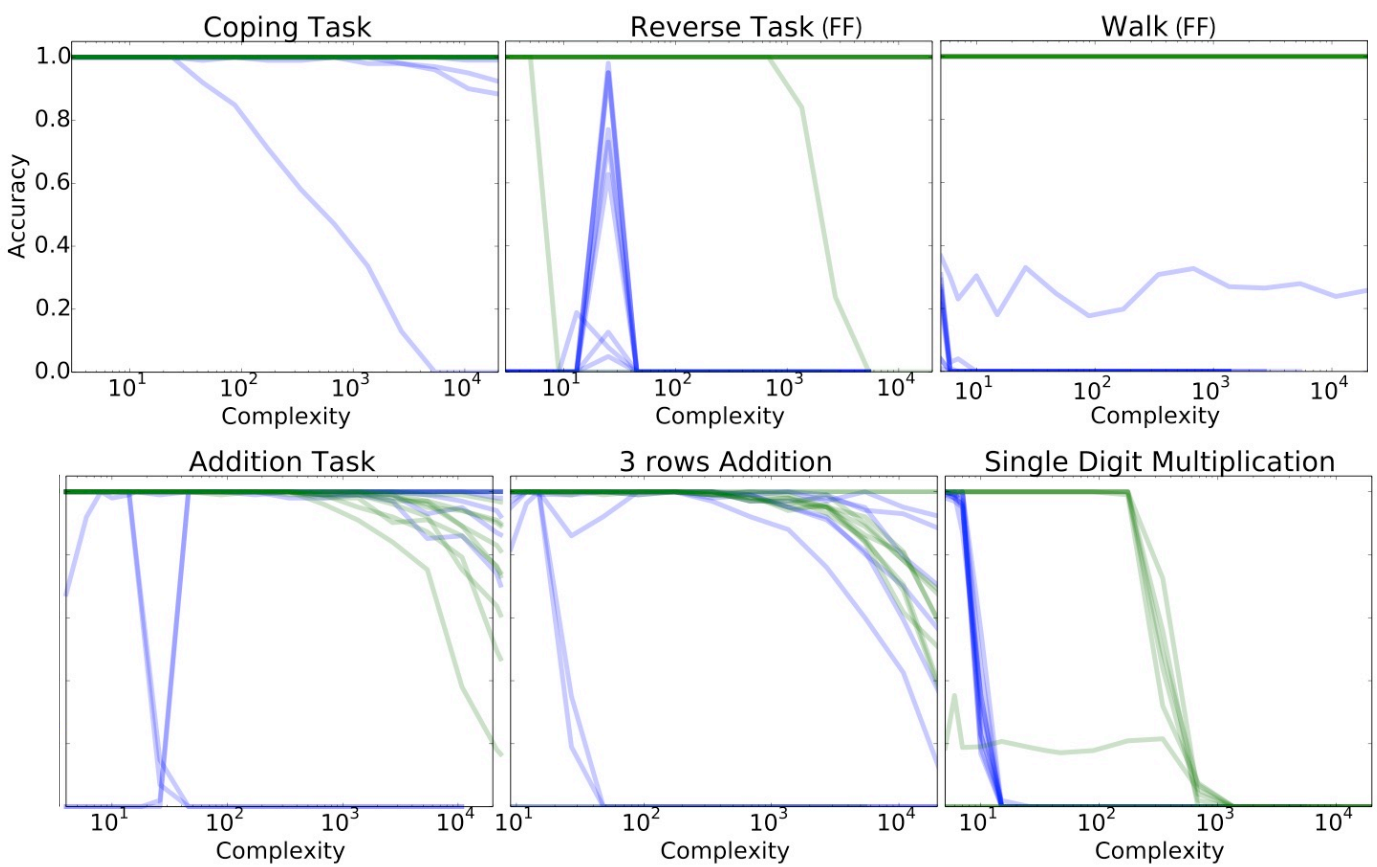
- Must learn both actions & symbol prediction
  - 0/1 Reward only after prediction of each digit
  - Abandon example as soon as mistake made

- Can't use backpropagation
  - We use Q-learning instead (with refinements)

Reinforcement Experiments

= Enhanced (all 3 terms)

= Regular Q-Learning



Related Work

- Neural Program Interpreters [Reed & deFreitas 2015]
  - More tasks, but supervised
- Neural Random-Access Machines [Kurach et al. 2015]

- Neural Turing Machine [Graves et al. 2014]
  - Continuous memory
- Reinforcement Learning NTM [Zaremba & Sutskever 2015]
  - Tapes as interfaces
- Program Induction, e.g. [Schmidhuber 2004]

Adding Interfaces to Deep Nets

- Often discrete in nature. What are the options?

- Continuous → use backprop
- Discrete → Use reinforcement learning

- Gumbel-Softmax trick
  - [The Concrete Distribution: a continuous relaxation of discrete random variables, Maddison et al., ICLR 2017]
  - [Categorical reparameterization by Gumbel-Softmax, Jang et al. ICLR 2017]
  - [GANs for sequences of discrete elements with the Gumbel-Softmax distribution, Kusner & Hernandez-Lobato, NIPS 2016 workshop]

Gumbel-Softmax Trick

- Reparameterization trick for discrete latent variables in stochastic nets
  - Analogous to Gaussian reparameterization in VAEs
- Sample according to:

$$g_i = -log(-log(U_i)) \qquad U_i \sim Unif[0,1]$$ Gumbel noise

$$y_i = \frac{\exp((\log(\pi_i) + g_i)/\tau)}{\sum_{j=1}^{k} \exp((\log(\pi_j) + g_j)/\tau)} \qquad \text{for } i = 1, ..., k.$$

Add log-props of each discrete category & pass through softmax

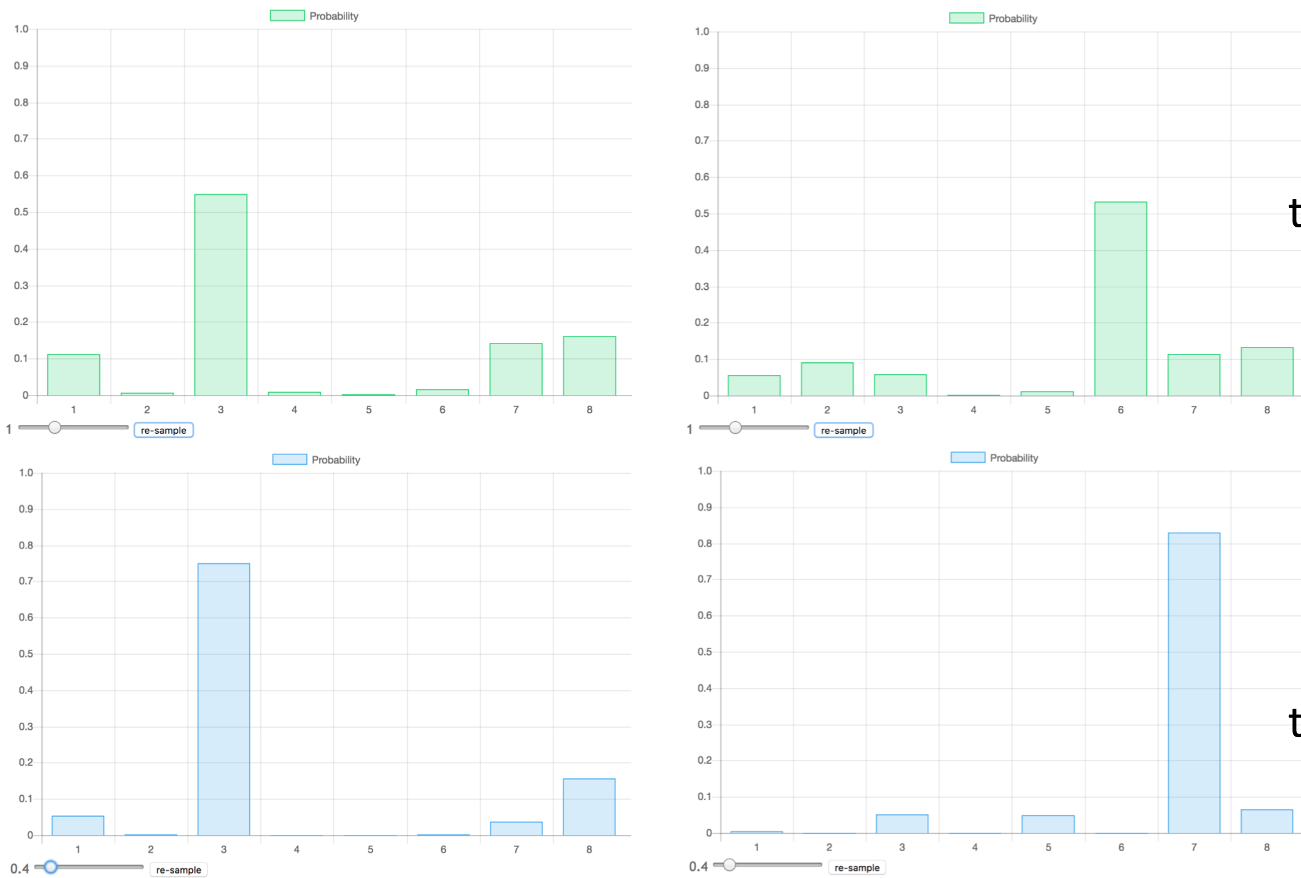
- Take sample from soft-max & b-prop as per usual
- Anneal temperature τ during training

https://blog.evjang.com/2016/11/tutorial-categorical-variational.html

Gumbel-Softmax Trick

- Samples from

$$y_i = \frac{\exp((\log(\pi_i) + g_i)/\tau)}{\sum_{j=1}^{k} \exp((\log(\pi_j) + g_j)/\tau)} \qquad \text{for } i = 1, \ldots, k.$$



Samples with temperature τ = 1.0

Samples with temperature τ = 0.4

https://blog.evjang.com/2016/11/tutorial-categorical-variational.html

Themes

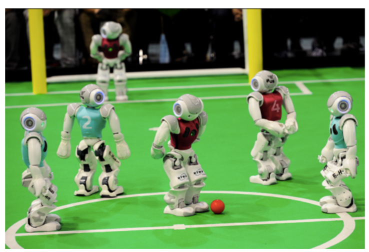
- 1. Memory in Deep Nets
- 2. Deep Nets for sets

What about set inputs?

Set structure

[PCMag]

[forbes.com]



{ “Sheep are afraid of wolves.”,
“Cats are afraid of dogs.”,
“Mice are afraid of cats.”,
“Gertrude is a sheep.”,
“What is Gertrude afraid of?”,
“Answer: wolves” }

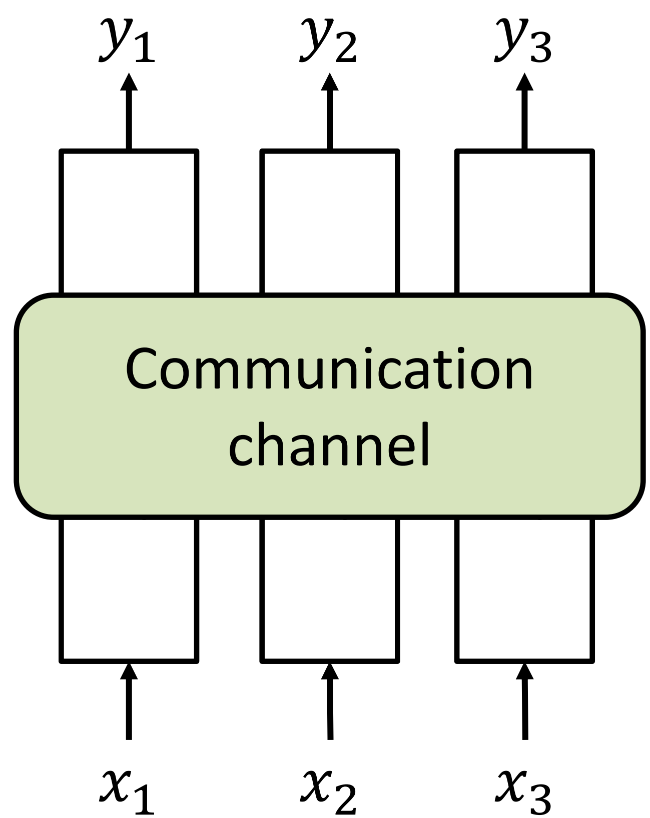
- Permutation invariance
- Dynamic sizing
- Single output
- Output for each element

?

Communication Neural Network (CommNet)

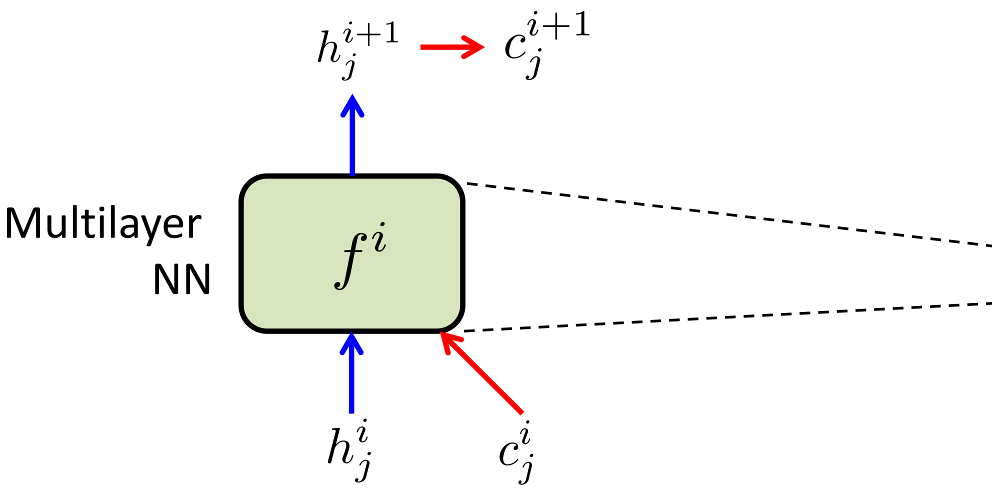
[Sukhbaatar et al. NIPS 2016]

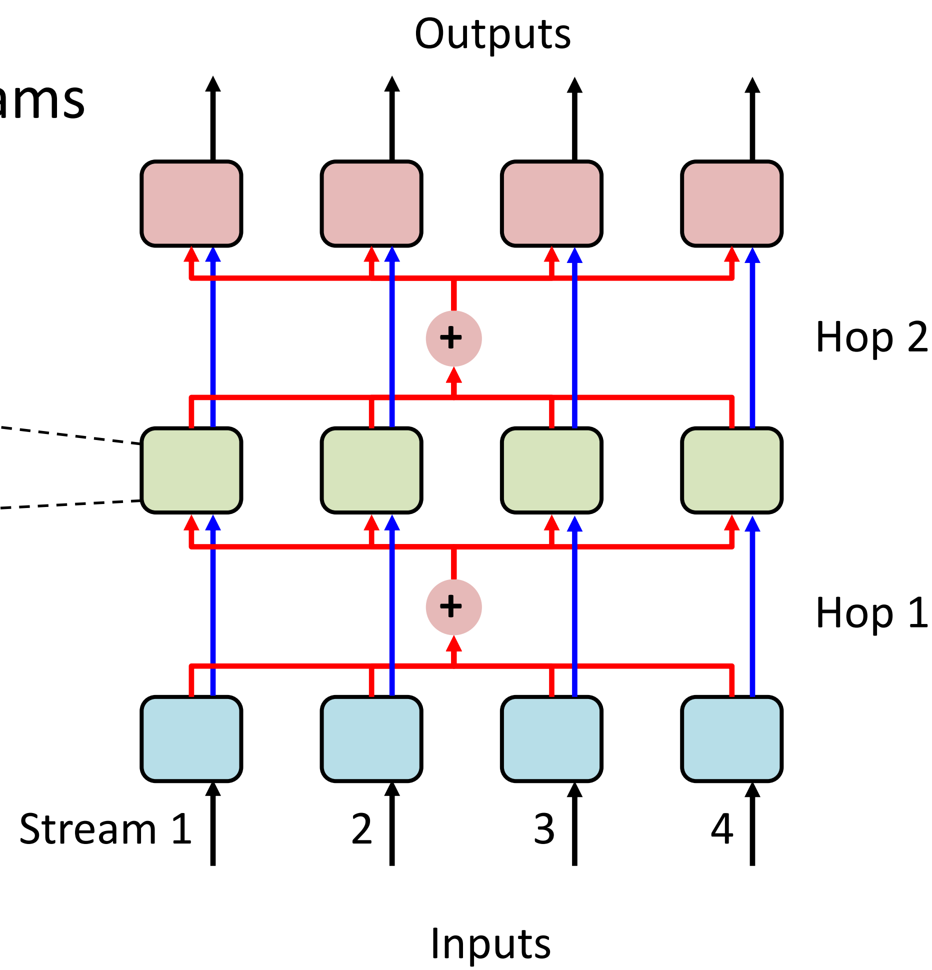
- Input and output is a **set**
- Each element has its own stream (weight shared)
- Distributed representation
- Continuous broadcast communication channel
- Streams must **learn to communicate** to solve task



CommNet Model

- J data points / streams
- # communication hops fixed
- Share parameters across streams





$$c_j^i = \frac{1}{J-1} \sum_{j' \neq j} h_{j'}^i$$

- Trained by backprop
- Invariant to order / number of inputs

[Sukhbaatar et al. NIPS 2016]

Module Structure

- Module f can be single/multi-layer NN or RNN/LSTM

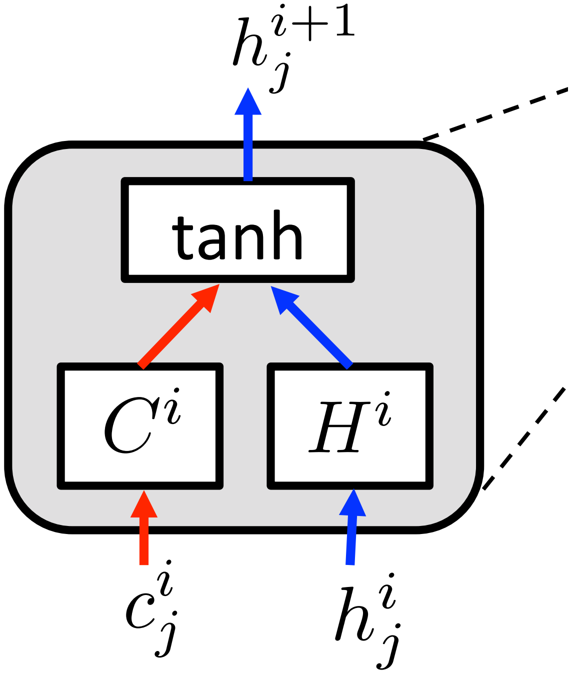
- At step i, two inputs:
  1. Hidden state vector h^i
  2. Communication vector c^i

- Output is new hidden state:

$$h_j^{i+1} = \sigma(\underbrace{H^i}_{\uparrow} h_j^i + \underbrace{C^i}_{\uparrow} c_j^i)$$

Learnable parameters

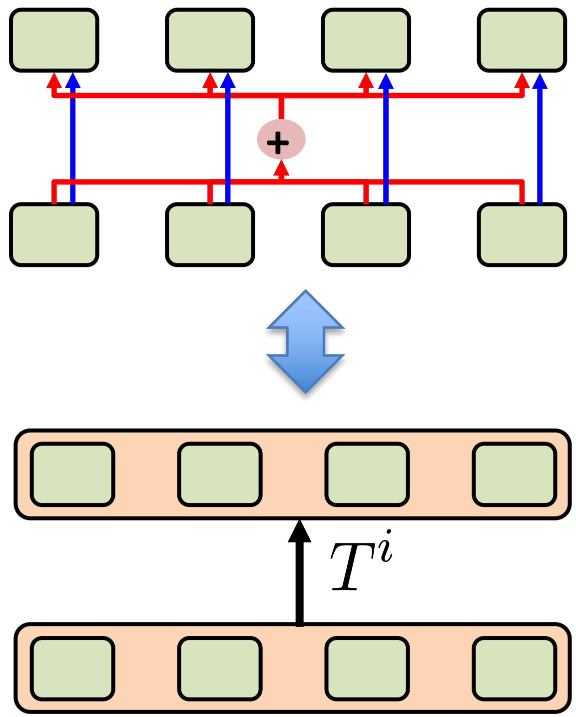
Module for agent j



[Sukhbaatar et al. NIPS 2016]

Big Model Interpretation

- Set of streams = one big model
- Let $\boxed{f}$ be single NN layer:



$$h_j^{i+1} = \sigma(H^i h_j^i + C^i c_j^i)$$

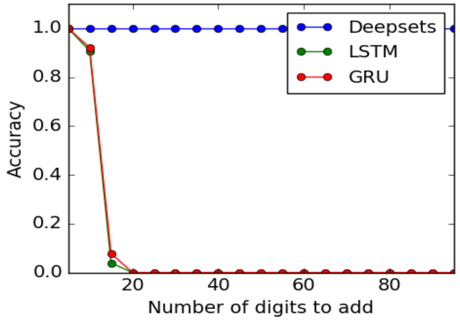
- N.B. Streams share parameters

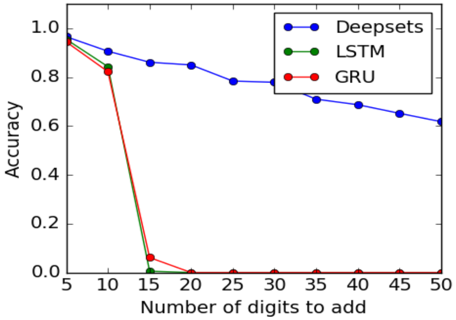
$$T^i = \begin{pmatrix} H^i & C^i & C^i & \dots & C^i \\ C^i & H^i & C^i & \dots & C^i \\ C^i & C^i & H^i & \dots & C^i \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ C^i & C^i & C^i & \dots & H^i \end{pmatrix}$$

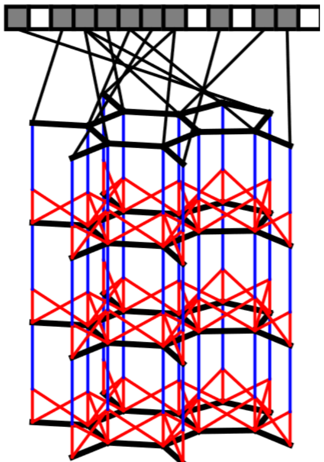
Dynamically sized: size of *T* can change depending on input set size

DeepSets [Zaheer et al., 2017]

- Architecture specialized to set input
- Make weights in each layer permutation invariant
  - Equal diagonal elements
  - Off-diagonal elements tied
- Stream for each element, summed in the end
- Experiment: Image with sample of 10 MNIST digits. Need to predict sum

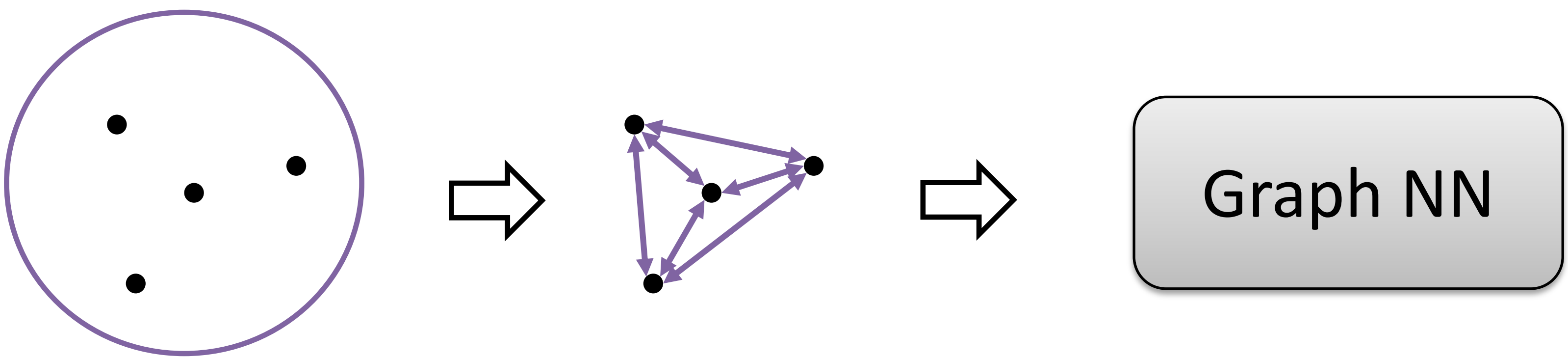






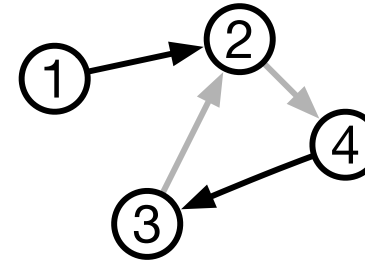
Graph NN view

- CommNet is a special case of Graph NN
- A set can be represented by a complete graph



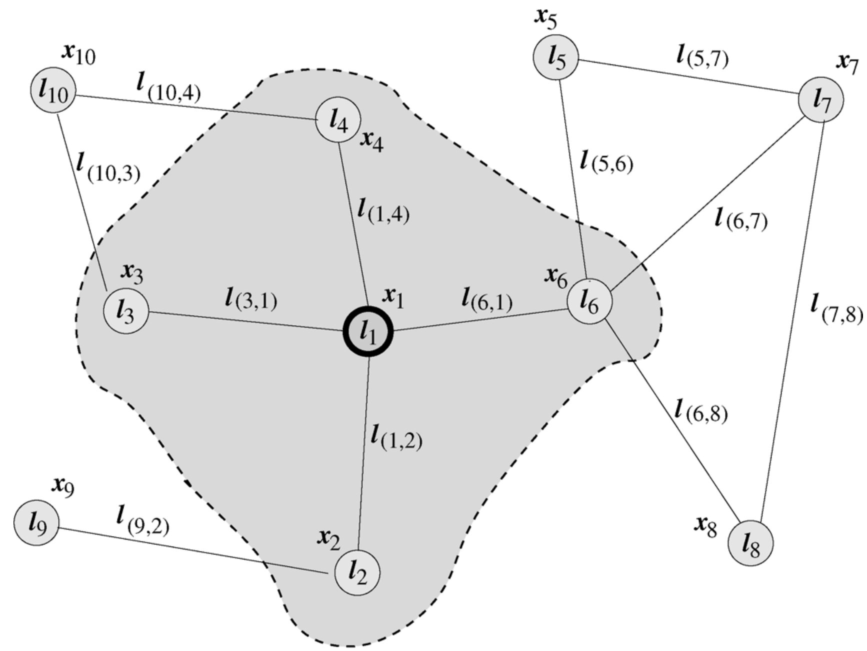
- Is everything Graph NN?

Graph NN



[Gori 2005, Scarselli 2009, Hamilton 2017]

- Nodes in a graph represent objects
- Edges represent their relationships.
- *State* of $\mathbf{x}_n$ each node n depends on neighborhood defined by graph



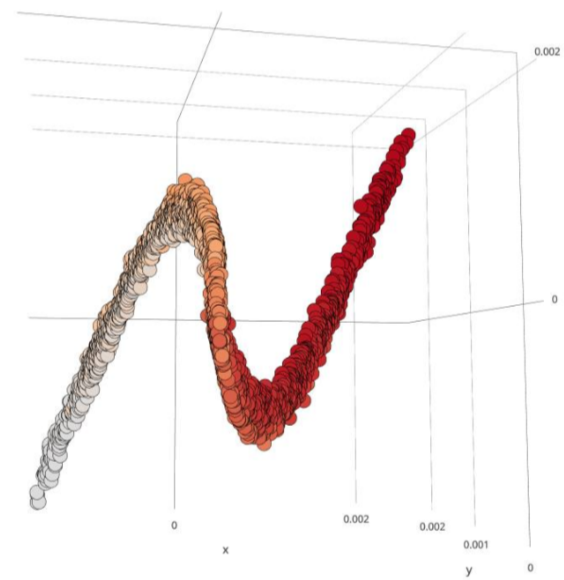
$$x_1 = f_w(\underbrace{l_1, l_{(1,2)}, l_{(3,1)}, l_{(1,4)}, l_{(6,1)}}_{l_{co[1]}}, \underbrace{x_2, x_3, x_4, x_6}_{x_{ne[1]}}, \underbrace{l_2, l_3, l_4, l_6}_{l_{ne[n]}})$$

- GNN for molecule [Duvenaud, 2015]
- Gated Graph NN [Li 2016, Bresson 2017]
- GNN+attention [Hoshen2017, Veličković2018]

Toy task

- Input = set of 5 numbers between 1 and 500
- Task: map the input to set of {1, 2, 3, 4, 5}

Model Φ	Training method: Supervised	Training method: Reinforcement
Independent	0.59	0.59
CommNet	**0.99**	**0.94**



[Sukhbaatar et al. NIPS 2016]

Experiment: bag to sequence

- Problem: given a set of words, arrange them in right order.

{is, mouse, cat, chasing} → "cat is chasing mouse"

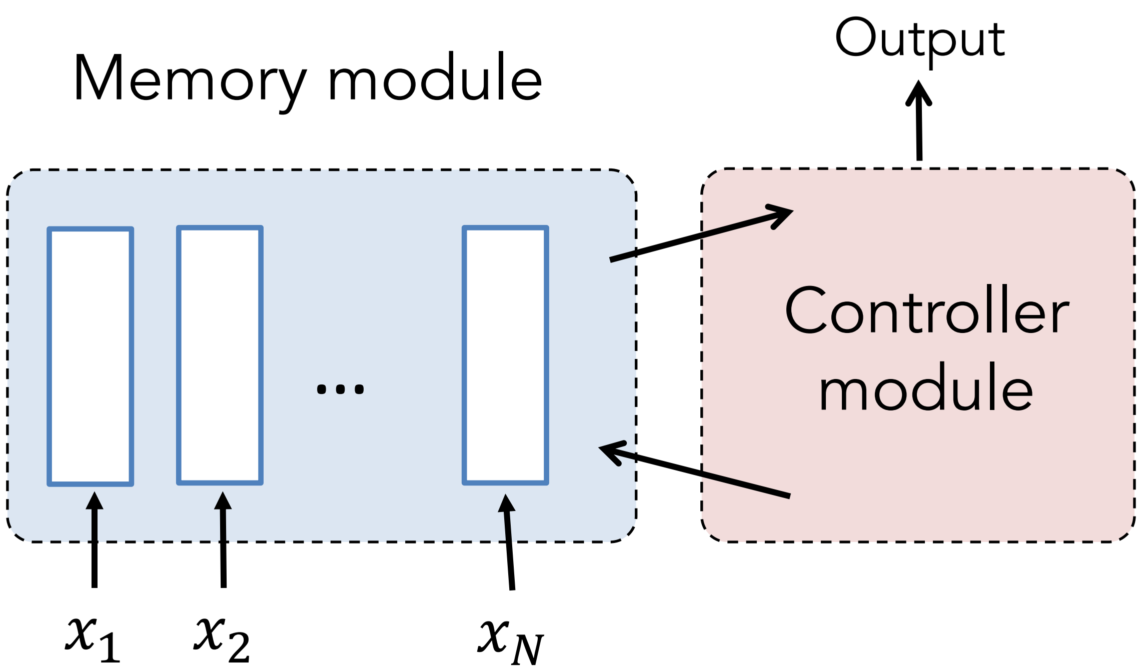
- Separate streams for each words
- After 2 hops, each stream output its location
- Data: Gigaword, 5 words, 2 layer MLP as f

	5-gram by KenLM	**Our model**
Error per word	40%	26%

[Sukhbaatar et al. NIPS 2016]

Another approach: Memory network

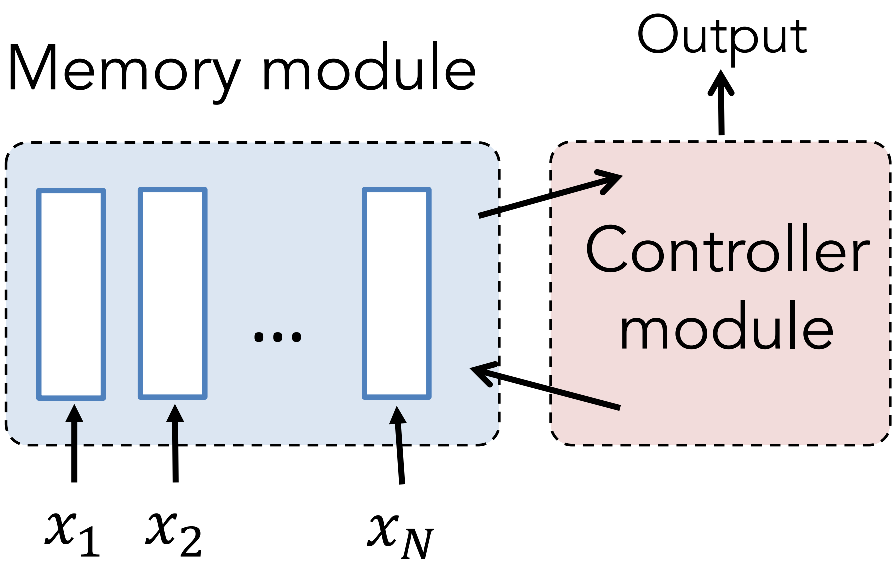
- Input is **set**, but single output
- Independently encode them → memory vectors
- Soft attention over memory vectors



MemNet VS CommNet

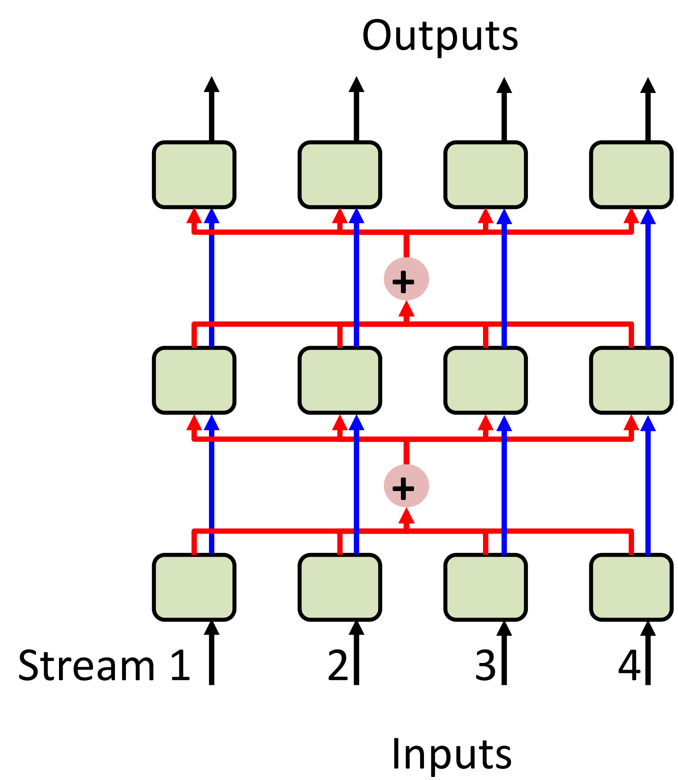
MemNet

- Central controller
- Serial processing



CommNet

- Distributed controller
- Parallel processing



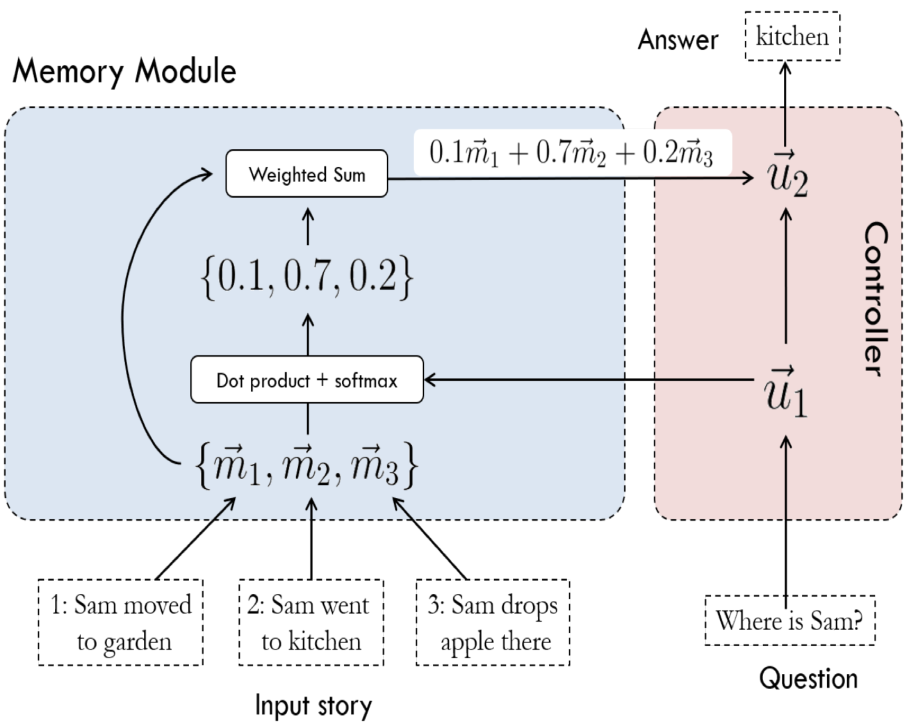
Experiment on bAbI Q&A data

- Data: 20 bAbI tasks [Weston et al. arXiv: 1502.05698, 2015]
- Answer questions after reading short story
- Small vocabulary, simple language
- Different tasks require different reasoning
- Training data size 1K or 10K for each task

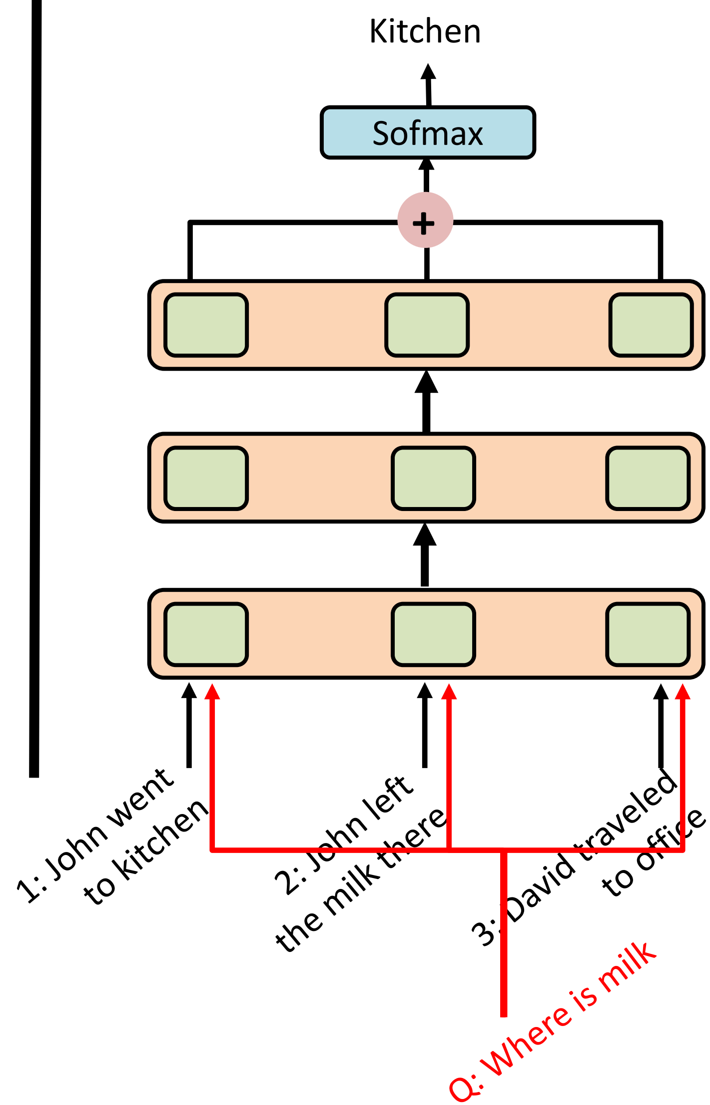
```
Sam walks into the kitchen.
Sam picks up an apple.
Sam walks into the bedroom.
Sam drops the apple.
Q: Where is the apple?
A. Bedroom
```

```
Brian is a lion.
Julius is a lion.
Julius is white.
Bernhard is green.
Q: What color is Brian?
A. White
```

MemNet



CommNet



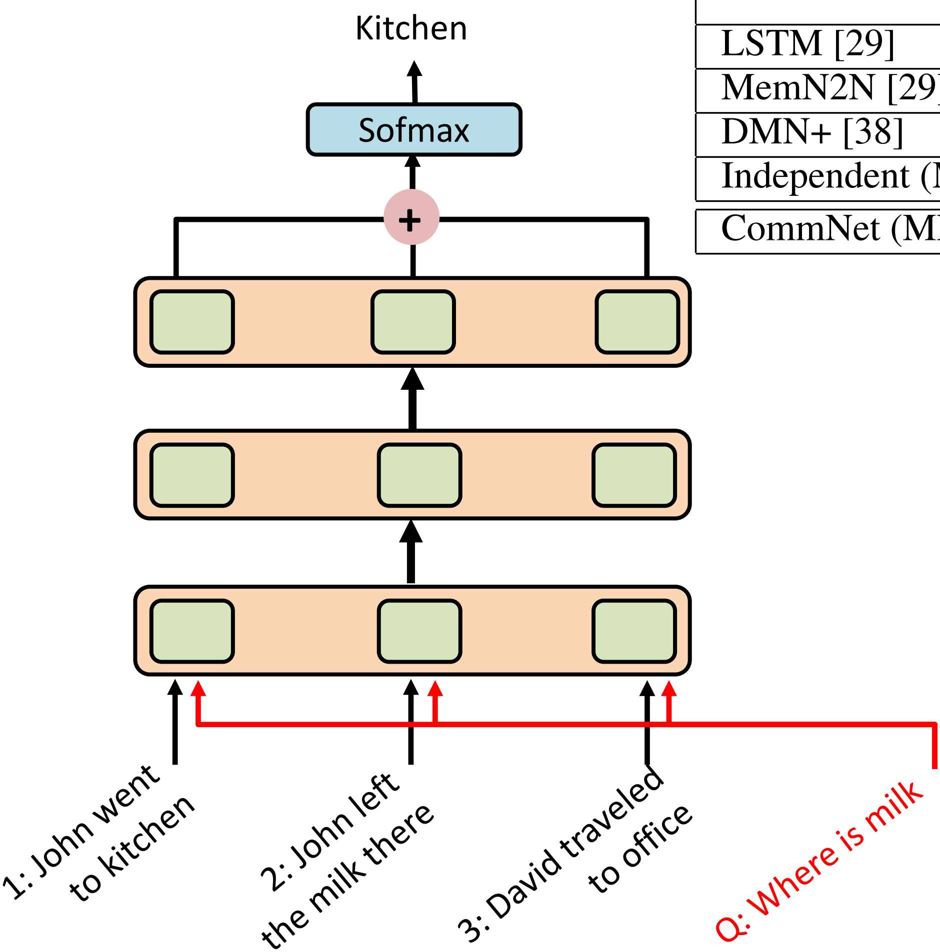
Examples of Attention Weights

- 2 test cases:

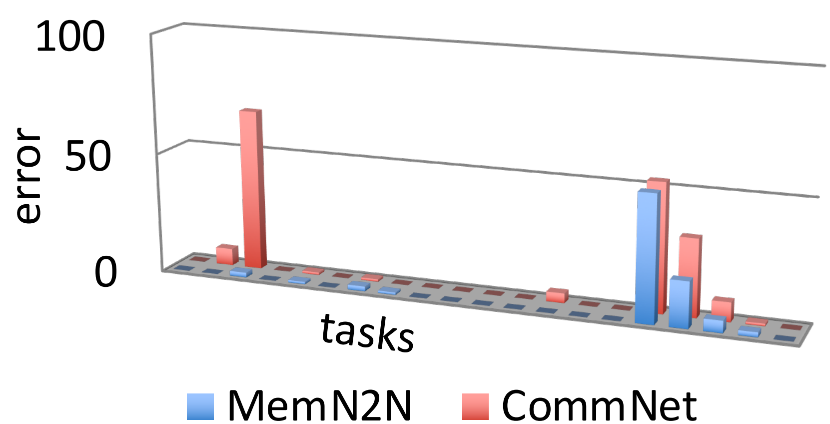
Story (2: 2 supporting facts)	**Hop 1**	**Hop 2**	**Hop 3**
John dropped the milk.	0.06	0.00	0.00
John took the milk there.	0.88	1.00	0.00
Sandra went back to the bathroom.	0.00	0.00	0.00
John moved to the hallway.	0.00	0.00	1.00
Mary went back to the bedroom.	0.00	0.00	0.00
Where is the milk? Answer: hallway Prediction: hallway			

Story (16: basic induction)	**Hop 1**	**Hop 2**	**Hop 3**
Brian is a frog.	0.00	0.98	0.00
Lily is gray.	0.07	0.00	0.00
Brian is yellow.	0.07	0.00	1.00
Julius is green.	0.06	0.00	0.00
Greg is a frog.	0.76	0.02	0.00
What color is Greg? Answer: yellow Prediction: yellow			

Experiment: 20 bAbI tasks

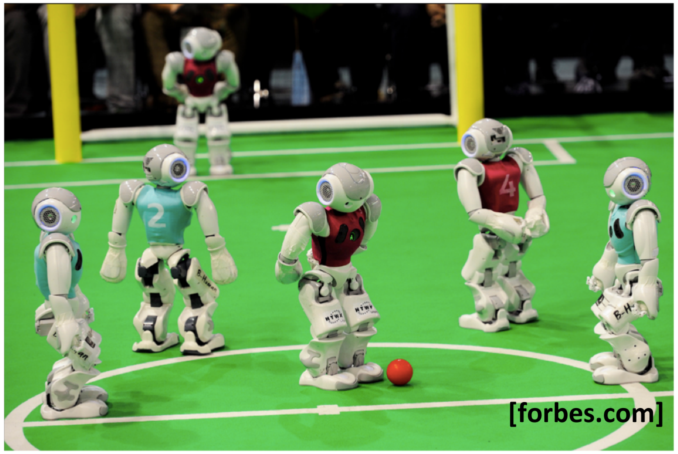


	Mean error (%)	Failed tasks (err. > 5%)
LSTM [29]	36.4	16
MemN2N [29]	4.2	3
DMN+ [38]	**2.8**	**1**
Independent (MLP module)	15.2	9
CommNet (MLP module)	7.1	3



Multi-agent communication for cooperative tasks

- Each agent can be view as an element of a set
- Communication doesn't have to be discrete symbols!

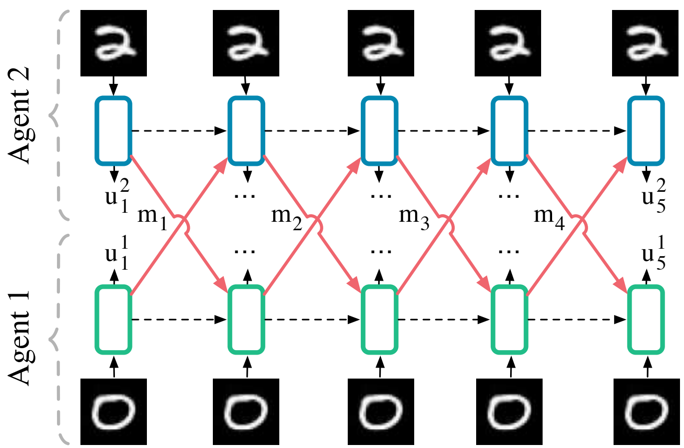


Related Work

- Multi-agent Reinforcement Learning
  - Lots of papers on collaborative task solving
  - But usually communication protocol fixed

- Recent/concurrent work:

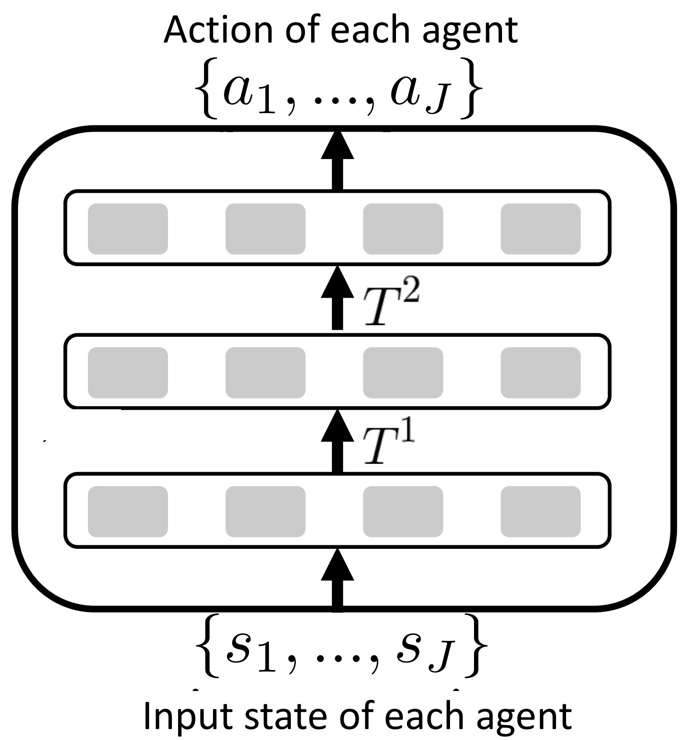
- **Learning to Communicate with Deep Multi-Agent Reinforcement Learning,** Jakob N. Foerster, Yannis M. Assael, Nando de Freitas, Shimon Whiteson, NIPS 2016



- **Emergence of Grounded Compositional Language in Multi-Agent Populations,** [Igor Mordatch](), [Pieter Abbeel](), arXiv 1703.04908
  - Uses Gumbel-Softmax trick

CommNet for Multiagent Reinforcement Learning (MARL)

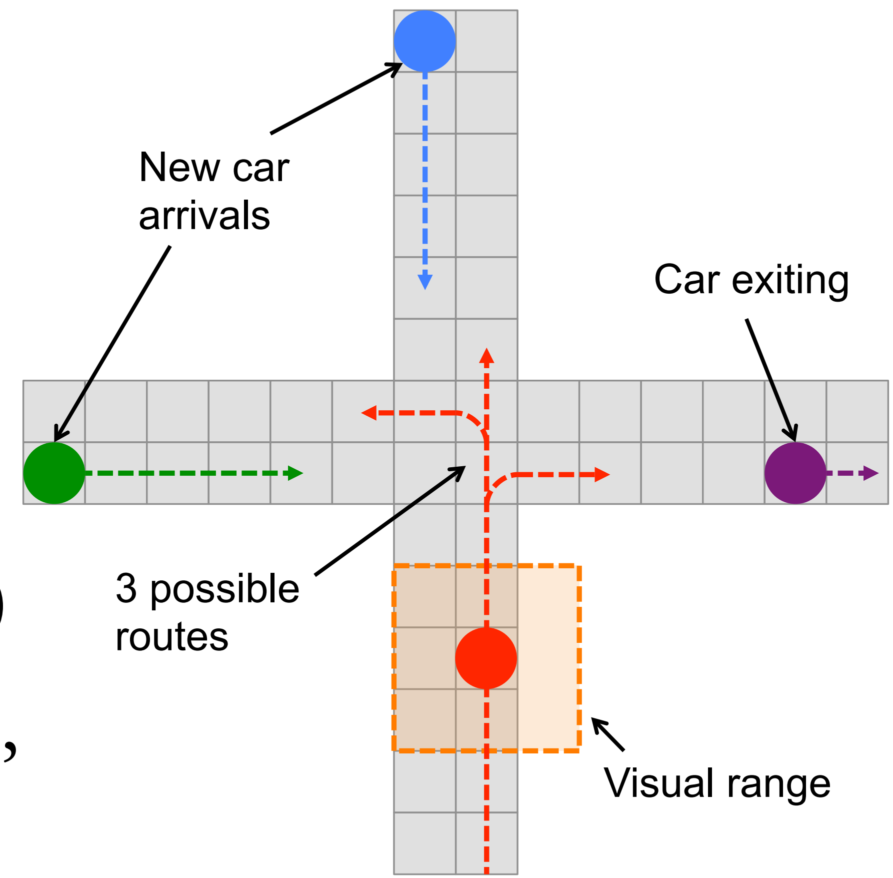
- Each stream is an agent
- Equal reward for all agents
- Agents collaborate to solve task
- Use Policy Gradient: REINFORCE [Williams et al. 1992]



[Sukhbaatar et al. NIPS 2016]

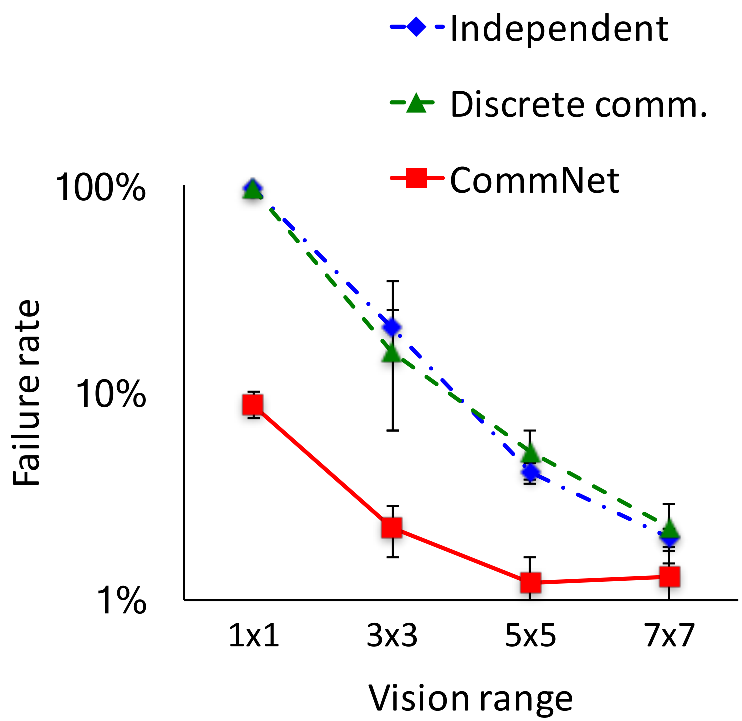
Traffic Junction game

- Cars on fixed routes
- Two actions: gas/brake
- Limited visibility
- Text representation
- Variable # cars (max 20)
- Rewards: collision = -10, delay = -0.01t



[Sukhbaatar et al. NIPS 2016]

Traffic Junction Results



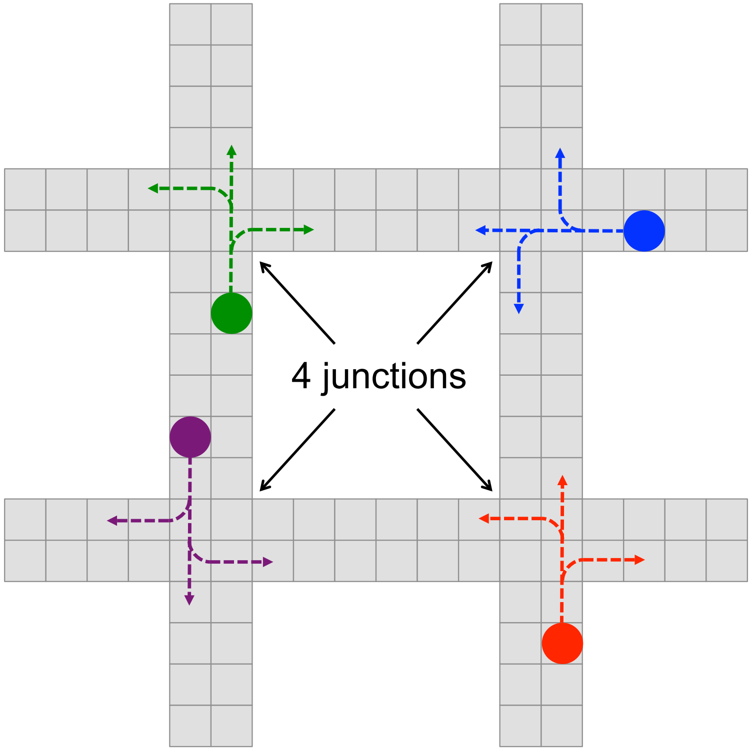
Failure rate

Model Φ	Module $f()$ type MLP	RNN	LSTM
Independent	20.6$\pm$ 14.1	19.5$\pm$ 4.5	9.4$\pm$ 5.6
Fully-connected	12.5$\pm$ 4.4	34.8$\pm$ 19.7	4.8$\pm$ 2.4
Discrete comm.	15.8$\pm$ 9.3	15.2$\pm$ 2.1	8.4$\pm$ 3.4
CommNet	**2.2$\pm$ 0.6**	**7.6$\pm$ 1.4**	**1.6$\pm$ 1.0**

[Sukhbaatar et al. NIPS 2016]

Traffic Junction (Hard version)

Communication type	Other game versions Easy (MLP)	Hard (RNN)
None	15.8± 12.5	26.9± 6.0
Discrete	1.1± 2.4	28.2± 5.7
Continuous	0.3± 0.1	22.5± 6.1
Cont. local	-	21.1± 3.4



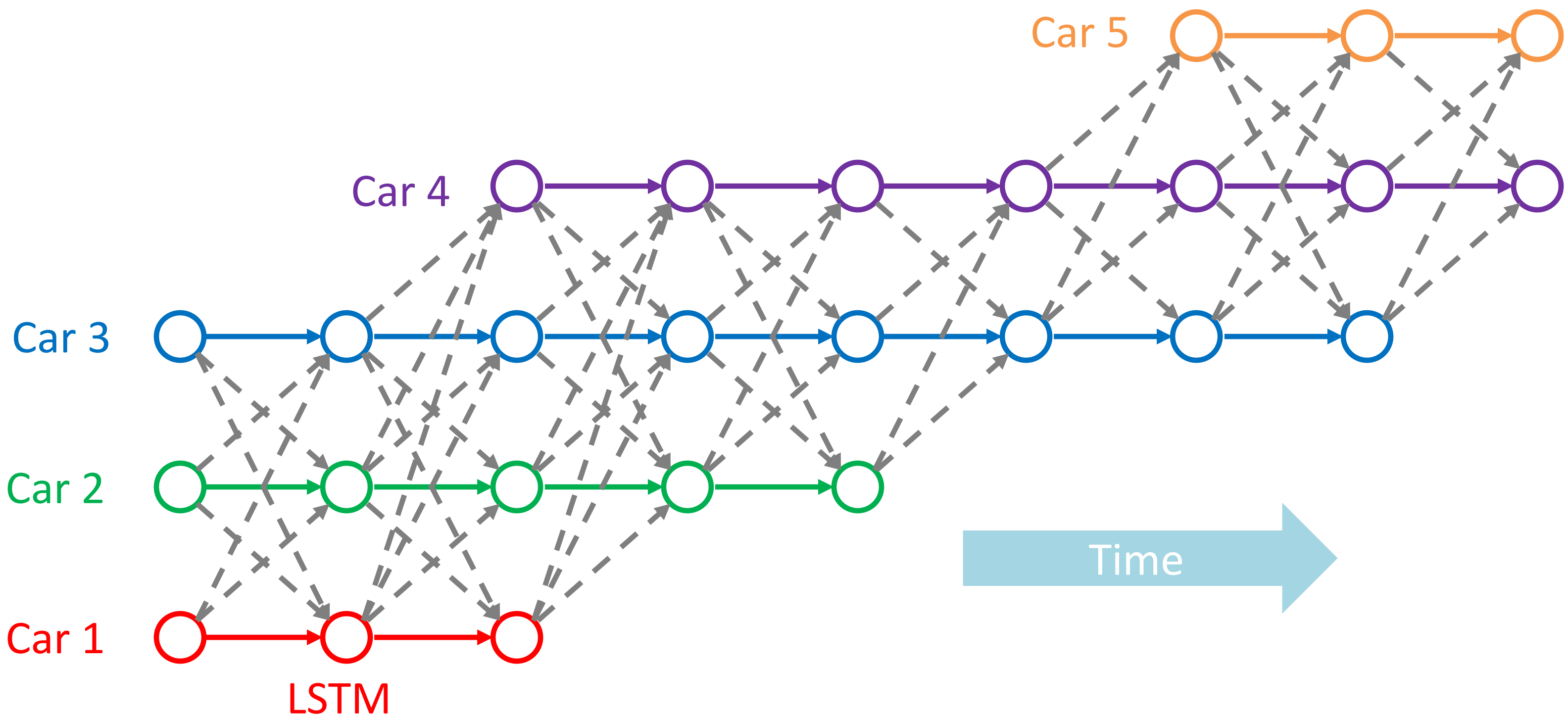
[Sukhbaatar et al. NIPS 2016]

Traffic Junction Movie

[Sukhbaatar et al. NIPS 2016]

Run time dynamic sizing

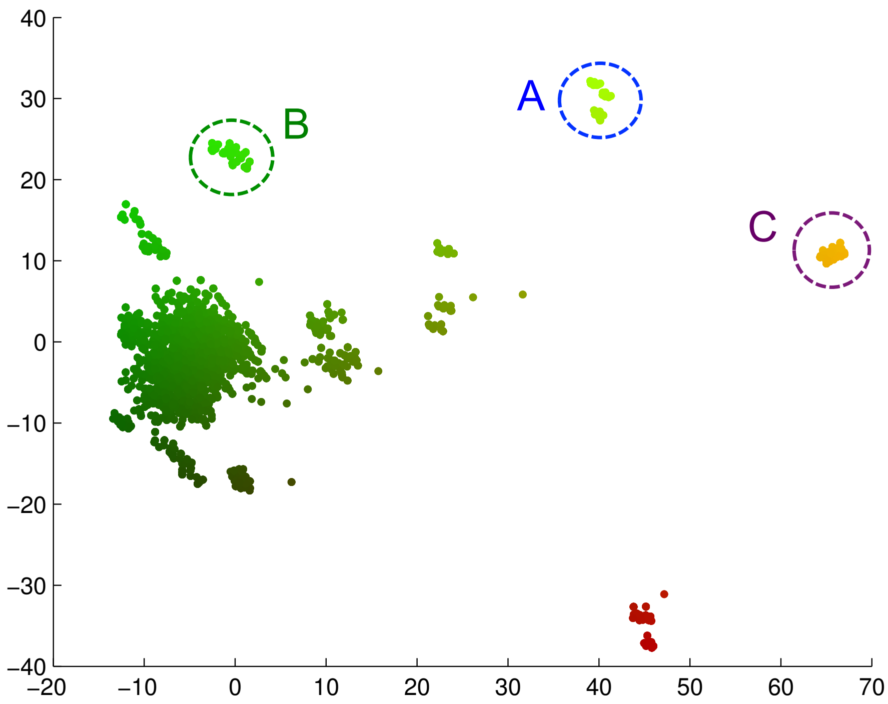
- The graph is changing with every layer



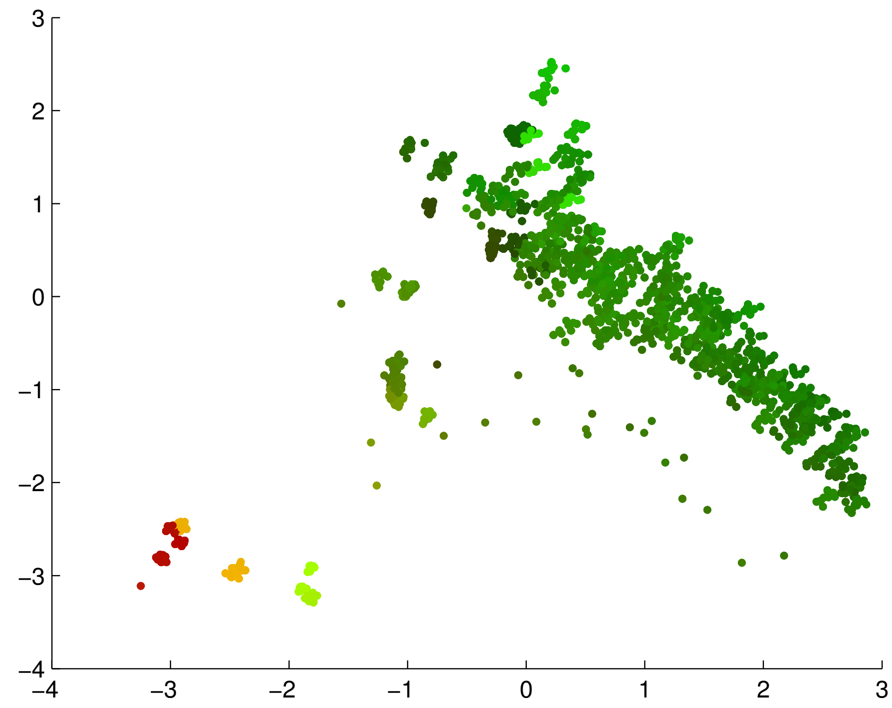
[Sukhbaatar et al. NIPS 2016]

How are the agents communicating?

PCA'd communication vectors



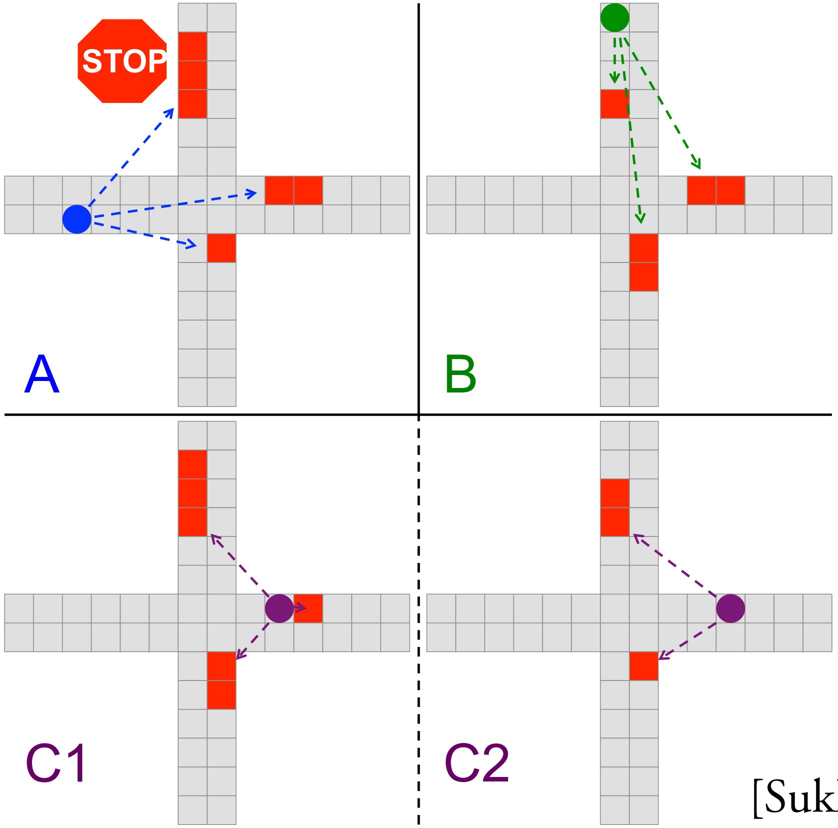
Corresponding hidden vectors



[Sukhbaatar et al. NIPS 2016]

How are the agents communicating?

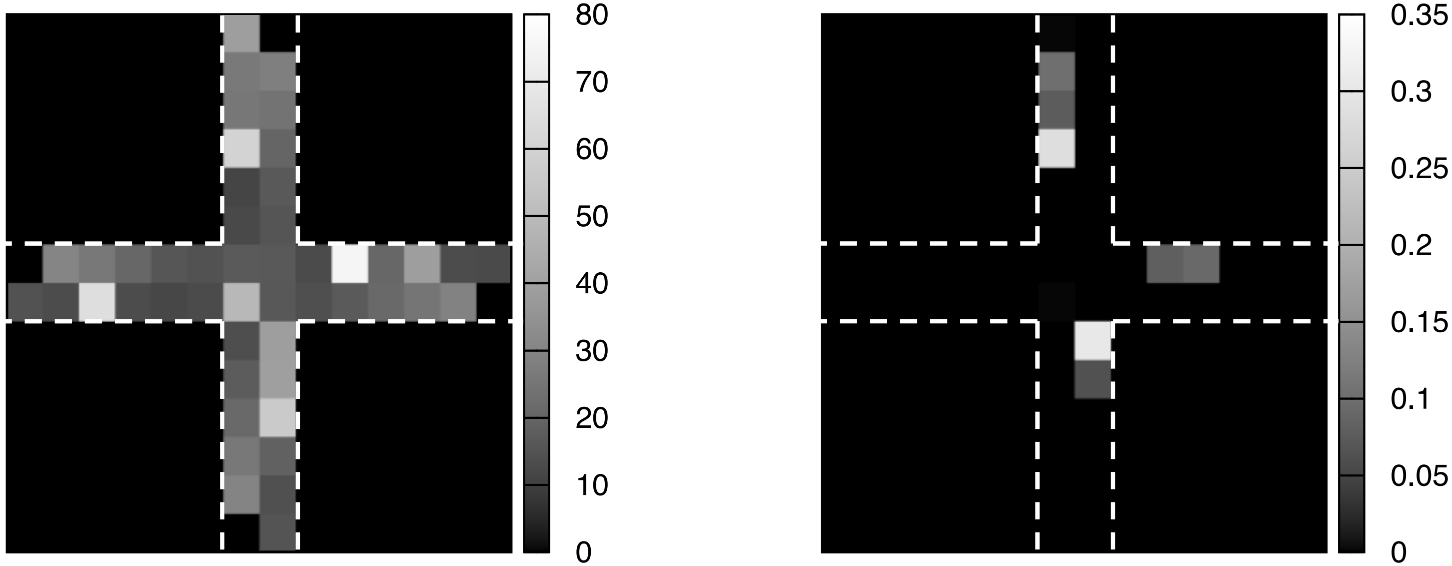
- Vectors from clusters correspond to distinct patterns of behavior:



[Sukhbaatar et al. NIPS 2016]

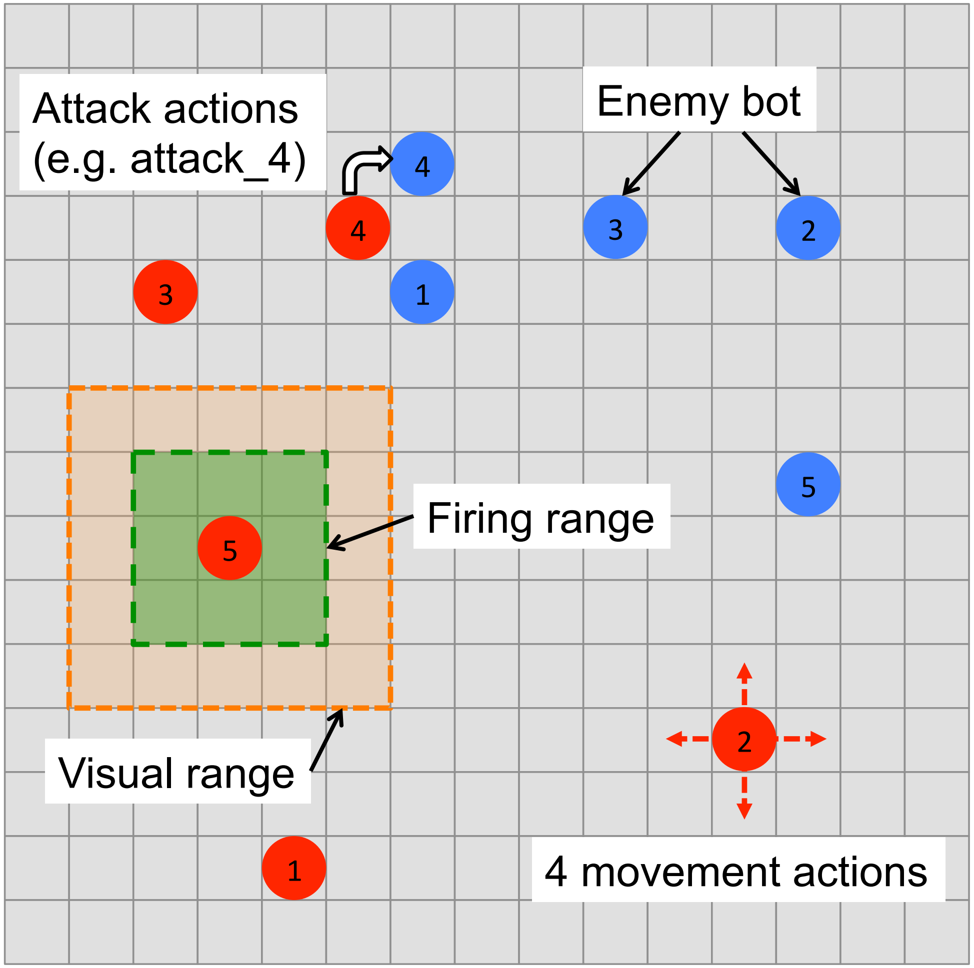
How are agents communicating?

- Average norm of the communication vectors and brake locations



[Sukhbaatar et al. NIPS 2016]

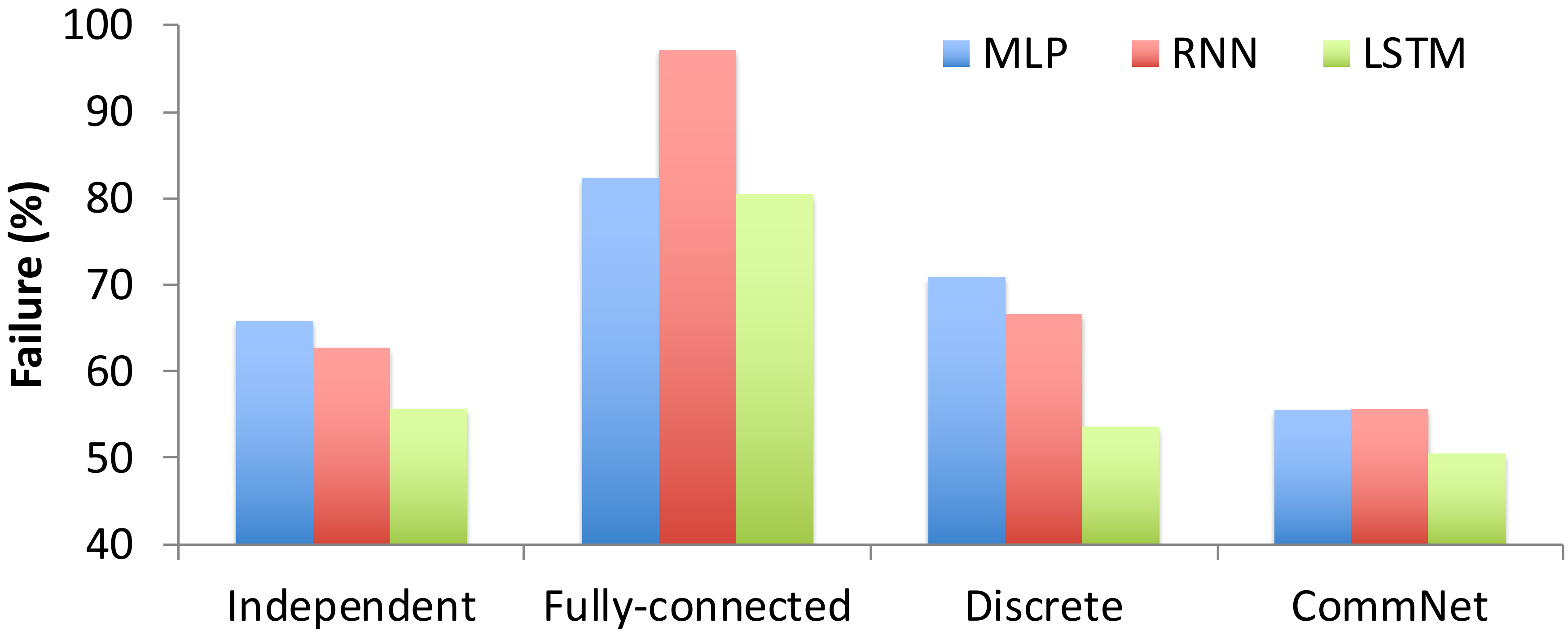
Combat game



[Sukhbaatar et al. NIPS 2016]

Experiment: Combat Game

- 5 agents vs 5 enemies in 15x15 map
- Health=3, Shot range=1, power=1, vision=1



[Sukhbaatar et al. NIPS 2016]

CommNN Summary

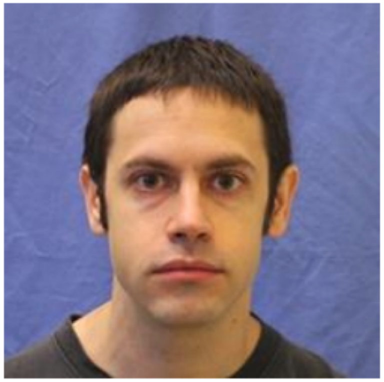
- Distributed NN model
  - Appropriate for tasks where input (and output) is set
- Models learn sparse communication protocol
- Can combine with RL for MARL problems
- Future directions
  - Generalize to non fully-cooperative setting
  - Which approach better? centralized or distributed

Thanks!

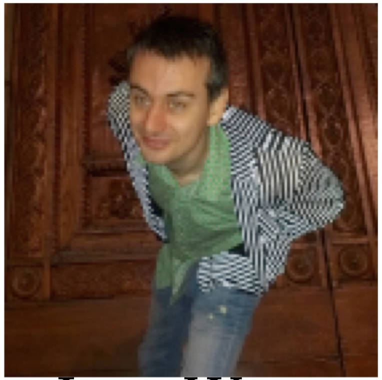
Facebook AI Research colleagues & NYU PhD students:

Sainbayar Sukhbaatar

Wojciech Zaremba



Arthur Szlam



Jason Weston

Tomas Mikolov

Armand Joulin

Memory References

- A. Graves. Generating sequences with recurrent neural networks. arXiv preprint: 1308.0850, 2013
- T. Mikolov, A. Joulin, S. Chopra, M. Mathieu, and M. A. Ranzato. Learning longer memory in recurrent neural networks. arXiv preprint:1412.7753, 2014
- A. Joulin, and T. Mikolov. Inferring Algorithmic Patterns with Stack-Augmented Recurrent Nets. arXiv preprint:1503.01007, 2015
- K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint:1409.1259, 2014
- J. Weston, S. Chopra, and A. Bordes. Memory networks. In International Conference on Learning Representations (ICLR), 2015
- S. Sukhbaatar, A. Szlam, J. Weston, and R. Fergus. End-To-End Memory Networks. arXiv preprint:1503.08895, 2015

References (I)

- J. Weston, S. Chopra, and A. Bordes. Memory networks. In International Conference on Learning Representations (ICLR), 2015
- J. Weston, A. Bordes, S. Chopra, and T. Mikolov. Towards AI-complete question answering: a set of prerequisite toy tasks. arXiv preprint: 1502.05698, 2015
- D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. ICLR, 2015
- K. Xu, J. L. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. ICML, 2015
- L. Yao, A. Torabi, K. Cho, N. Ballas, C. Pal, H. Larochelle, and A. Courville. Describing videos by exploiting temporal structure. ICCV, 2015
- J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio. Attention-based models for speech recognition. NIPS, 2015

References (II)

- O. Vinyals, M. Fortunato, and N. Jaitly. Pointer networks. NIPS, 2015
- M. C. Mozer and S. Das. A connectionist symbol manipulator that discovers the structure of context-free languages. NIPS, 1993
- A. Joulin, and T. Mikolov. Inferring Algorithmic Patterns with Stack-Augmented Recurrent Nets. NIPS, 2015
- A. Graves, G. Wayne, and I. Danihelka. Neural Turing Machines. arXiv preprint: 1410.5401, 2014
- A. Kumar, O. Irsoy, P. Ondruska, M. Iyyer, J. Bradbury, I. Gulrajani, and R. Socher. Ask Me Anything: Dynamic Memory Networks for Natural Language Processing. arXiv preprint: 1506.07285, 2015
- K. Hermann, T. Kočiský, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom. Teaching Machines to Read and Comprehend, arXiv preprint: 1506.03340, 2015