NEW YORK UNIVERSITY

Facebook AI Research

Learning Communication and Abstraction with Neural Nets

Rob Fergus

Facebook AI Research, New York University

Work with:

Sainbayar Sukhbaatar

Arthur Szlam

Overview

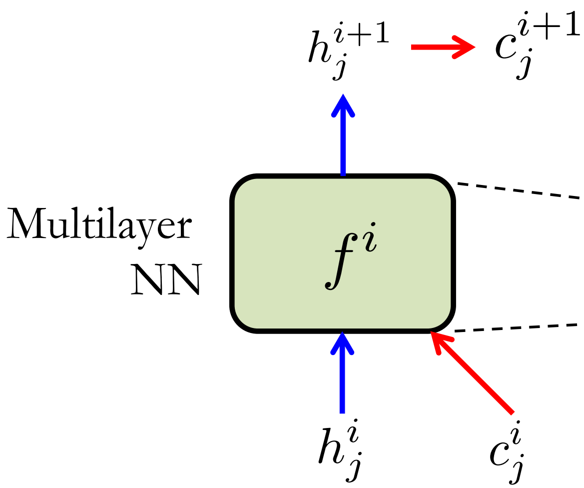
- Communication
  - Learnable communication protocol between neural net agents solving collaborative tasks
- Abstraction (ongoing work)
  - Hierarchy of actions in reinforcement learning for better scaling, planning and exploration

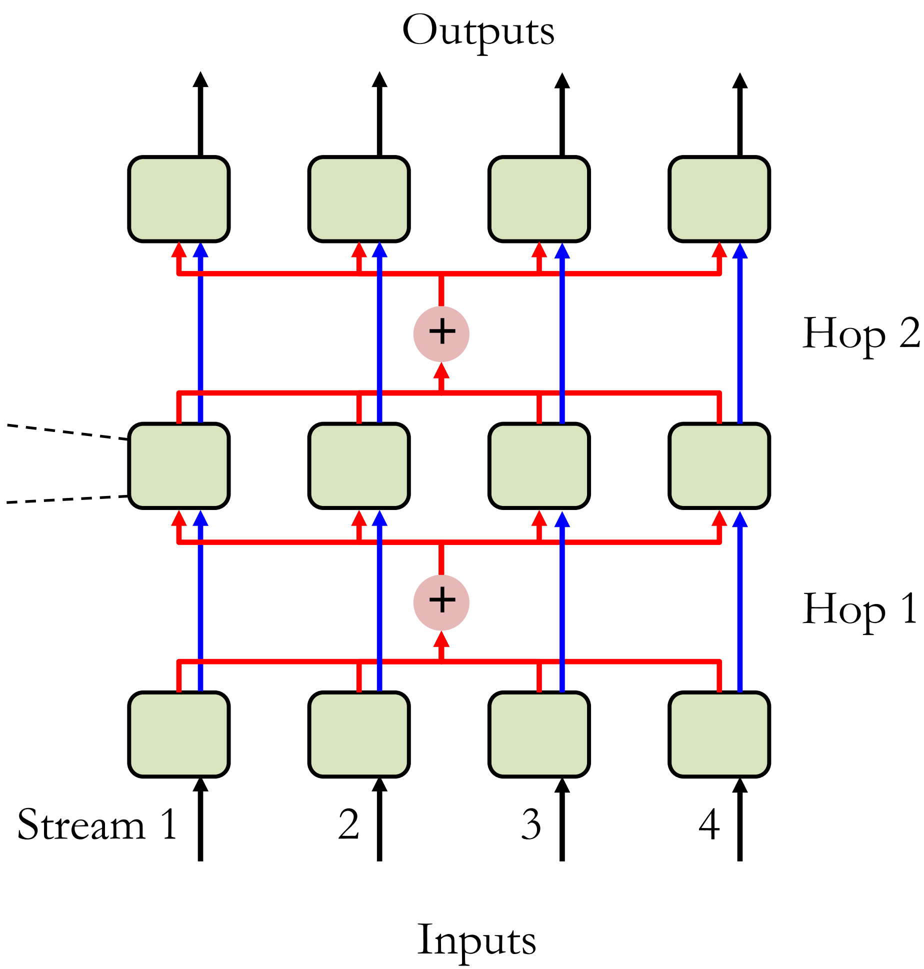
Communication Neural Network (CommNet)

- Input is a **set**
- Each element has its own processing stream
- Continuous broadcast communication channel between streams
- Streams must **learn to communicate** to solve task

CommNet Model

- J data points / streams
- # communication hops fixed





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

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

Module Structure

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

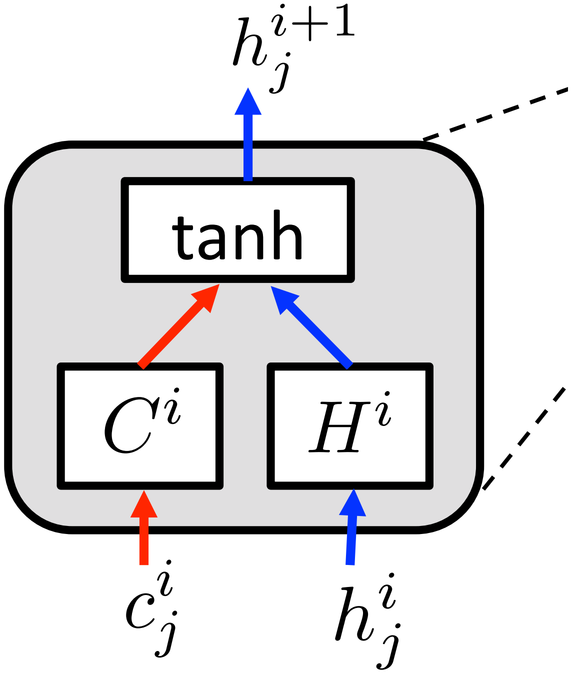
- At step i, two inputs:
  1. Hidden state vector h^i
  2. Communication vector c^i

- Output is new hidden state:

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

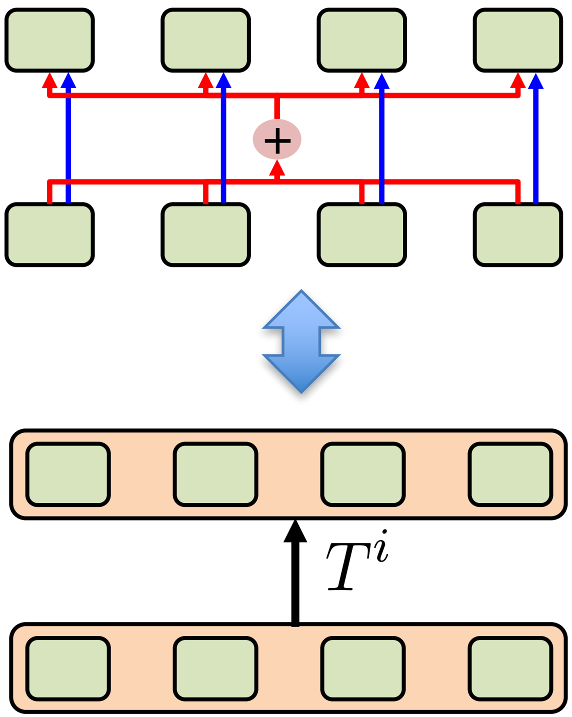
Learnable parameters

Module for agent j



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

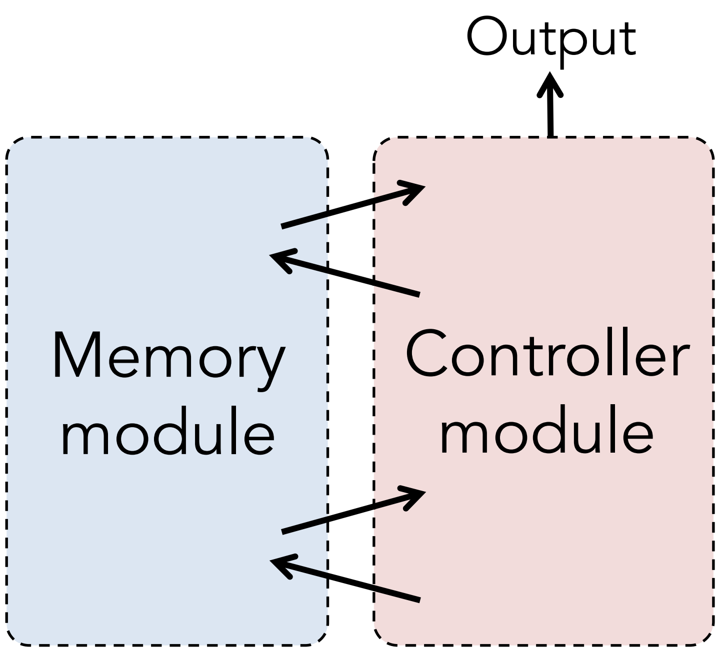
$$T^i = \begin{pmatrix} H^i & C^i & C^i & \dots & C^i \\ C^i & H^i & C^i & \dots & C^i \\ C^i & C^i & H^i & \dots & C^i \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ C^i & C^i & C^i & \dots & H^i \end{pmatrix}$$

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

CommNet vs External Memory

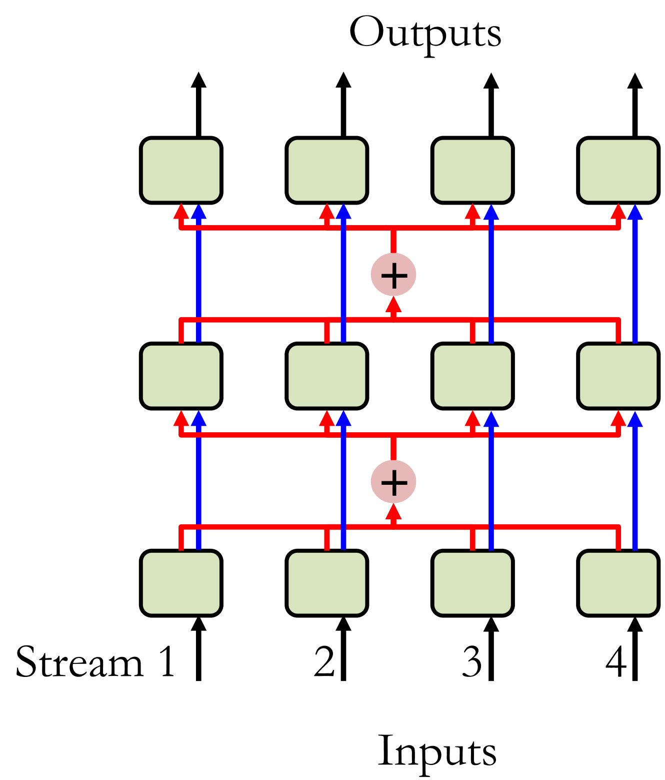
NTM / MemNet

- Separate controller
- Serial processing

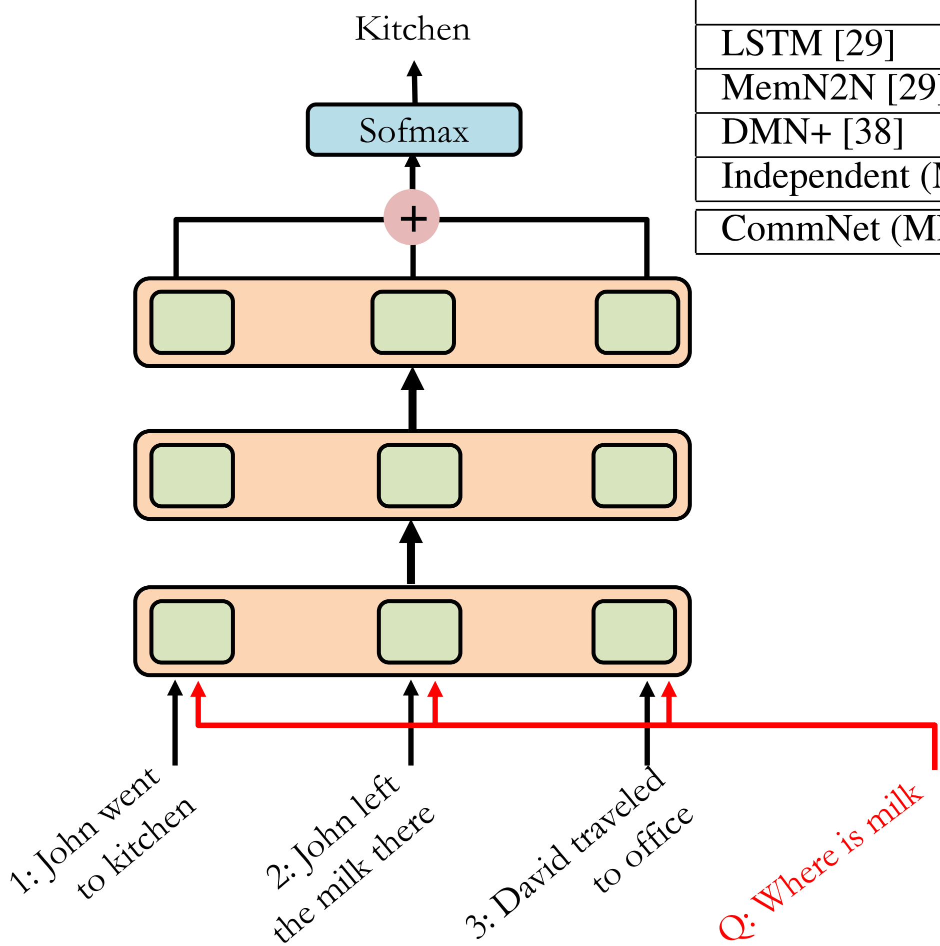


CommNet

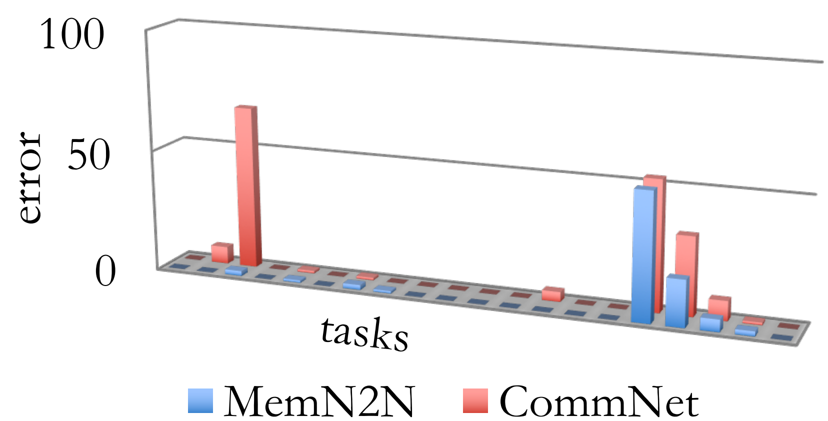
- Distributed controller
- Parallel processing



Experiment: 20 bAbI tasks

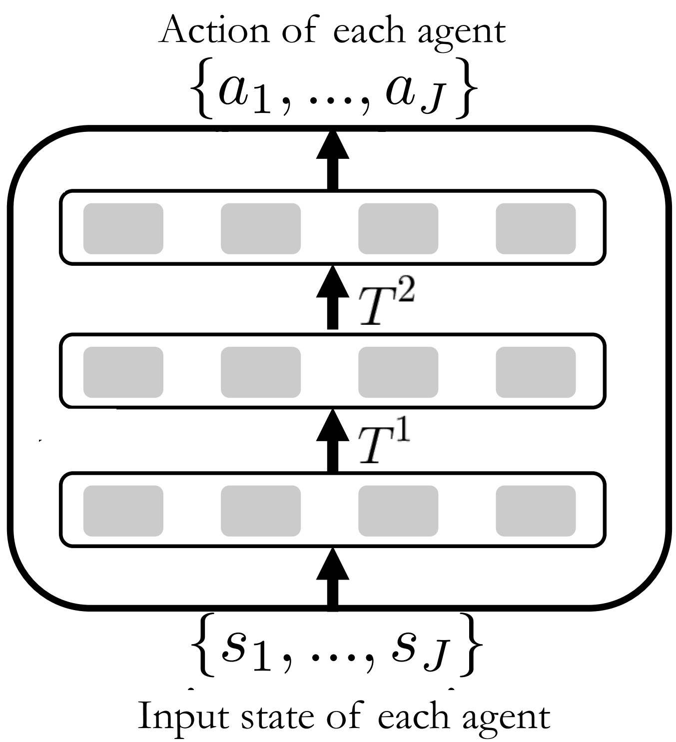


	Mean error (%)	Failed tasks (err. > 5%)
LSTM [29]	36.4	16
MemN2N [29]	4.2	3
DMN+ [38]	**2.8**	**1**
Independent (MLP module)	15.2	9
CommNet (MLP module)	7.1	3



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]



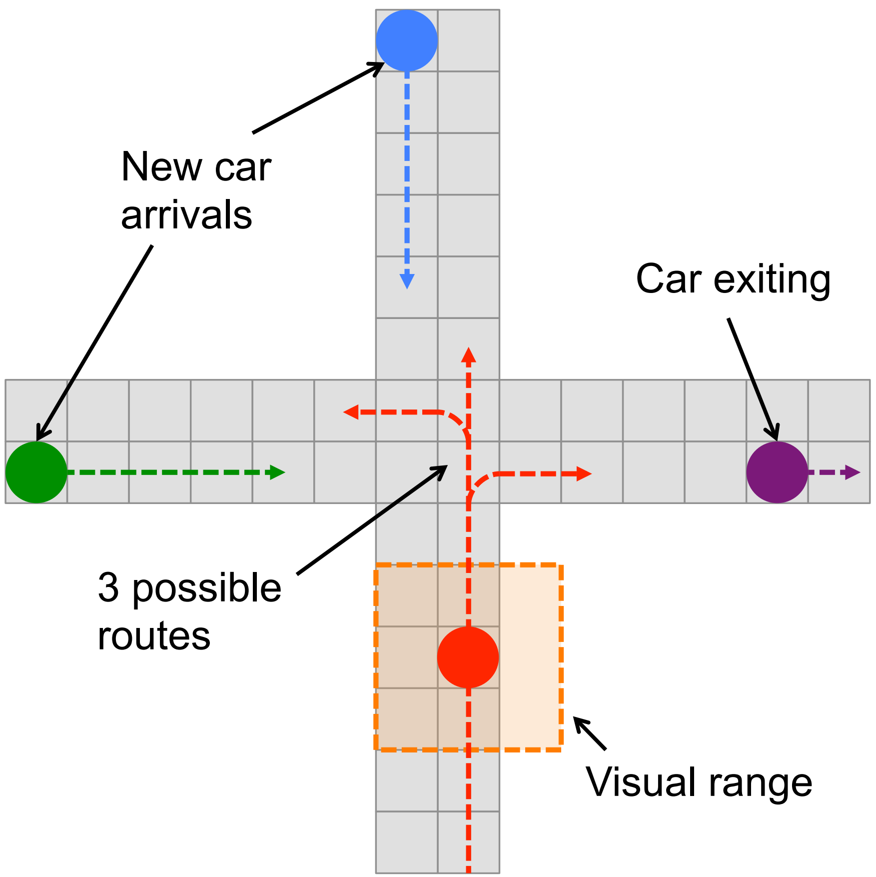
Traffic Junction game

- Cars on fixed routes
- Two actions: gas/brake
- Limited visibility
- Text representation
- Variable # cars (max 20)

- Rewards:

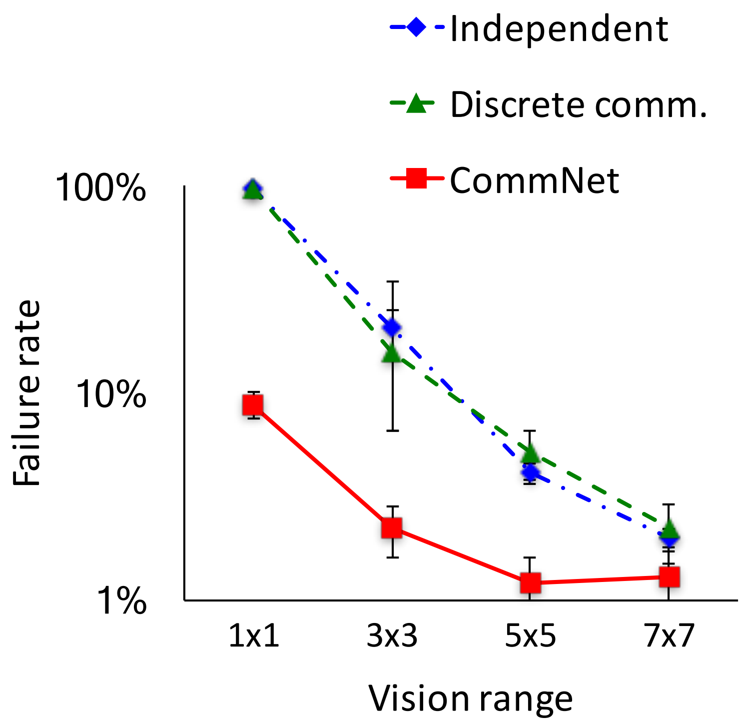
  Collision = -10

  Delay = -0.01t



Traffic Junction Movie

Traffic Junction Results

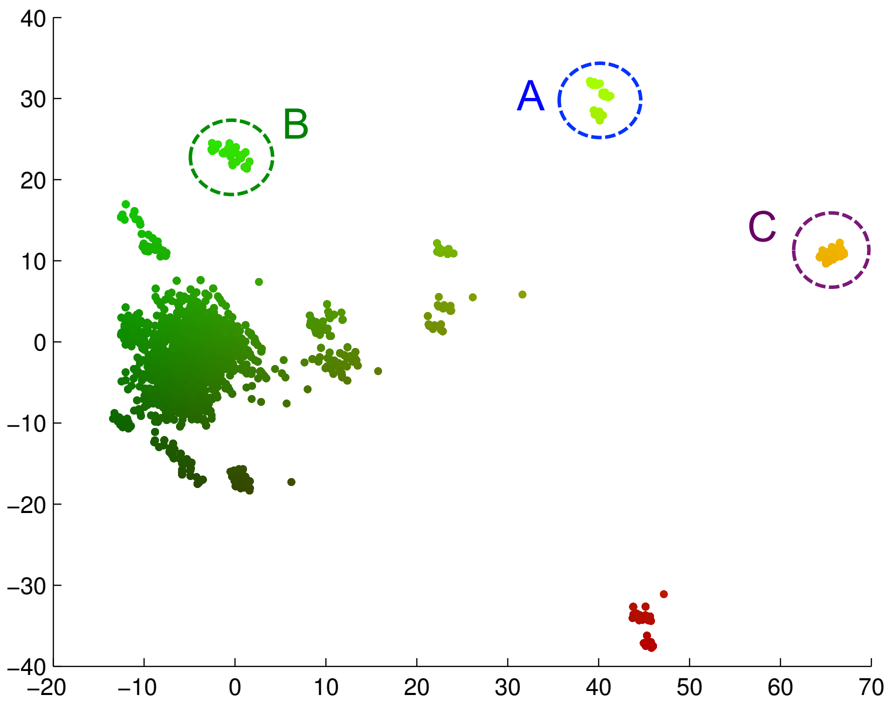


Failure rate

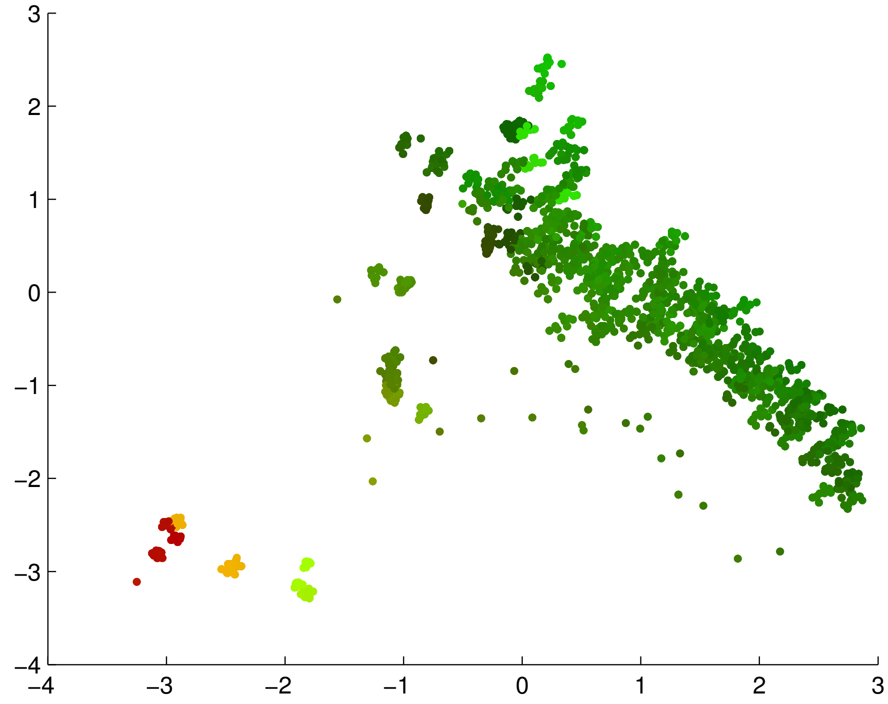
Model Φ	Module $f()$ type MLP	RNN	LSTM
Independent	20.6± 14.1	19.5± 4.5	9.4± 5.6
Fully-connected	12.5± 4.4	34.8± 19.7	4.8± 2.4
Discrete comm.	15.8± 9.3	15.2± 2.1	8.4± 3.4
CommNet	**2.2± 0.6**	**7.6± 1.4**	**1.6± 1.0**

How are the agents communicating?

PCA'd communication vectors

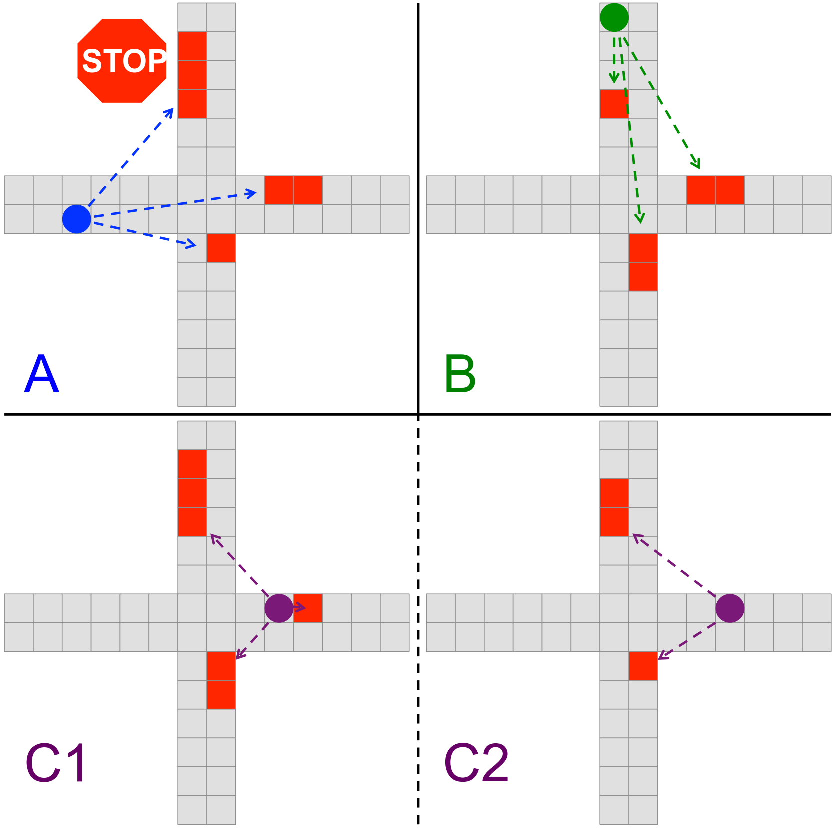


Corresponding hidden vectors

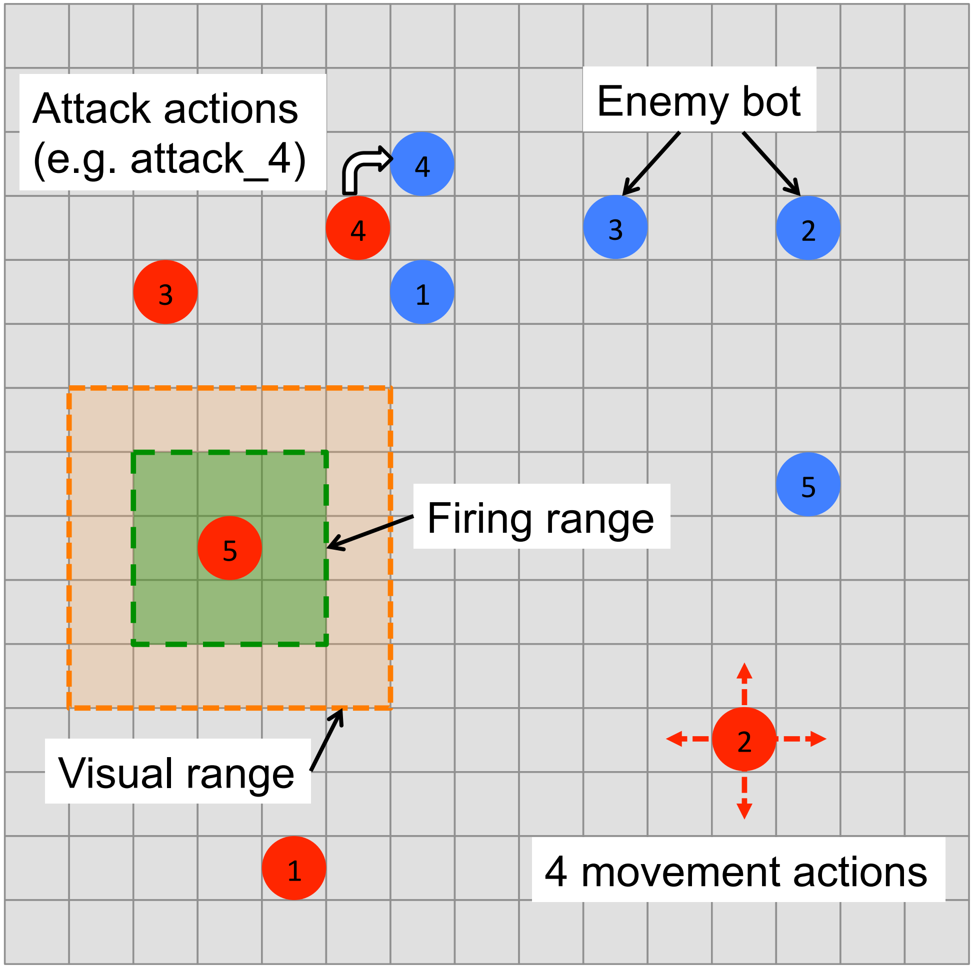


How are the agents communicating?

- Vectors from clusters correspond to distinct patterns of behavior:



Combat game


Attack actions
(e.g. attack_4)
Enemy bot
4
4
3
2
3
1
5
5
Firing range
2
Visual range
1
4 movement actions

Experiment: Combat Game

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

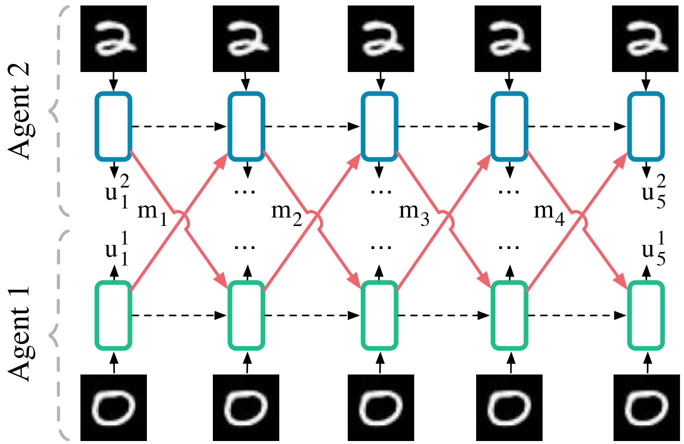
Model Φ	Module $f()$ type MLP	RNN	LSTM
Independent	34.2± 1.3	37.3± 4.6	44.3± 0.4
Fully-connected	17.7± 7.1	2.9± 1.8	19.6± 4.2
Discrete comm.	29.1± 6.7	33.4± 9.4	46.4± 0.7
CommNet	**44.5**± **13.4**	**44.4**± **11.9**	**49.5**± **12.6**

Model Φ	Other game variations (MLP) $m = 3$	$m = 10$	5×5 vision
Independent	29.2± 5.9	30.5± 8.7	60.5± 2.1
CommNet	**51.0**± **14.1**	**45.4**± **12.4**	**73.0**± **0.7**

Related Work (I)

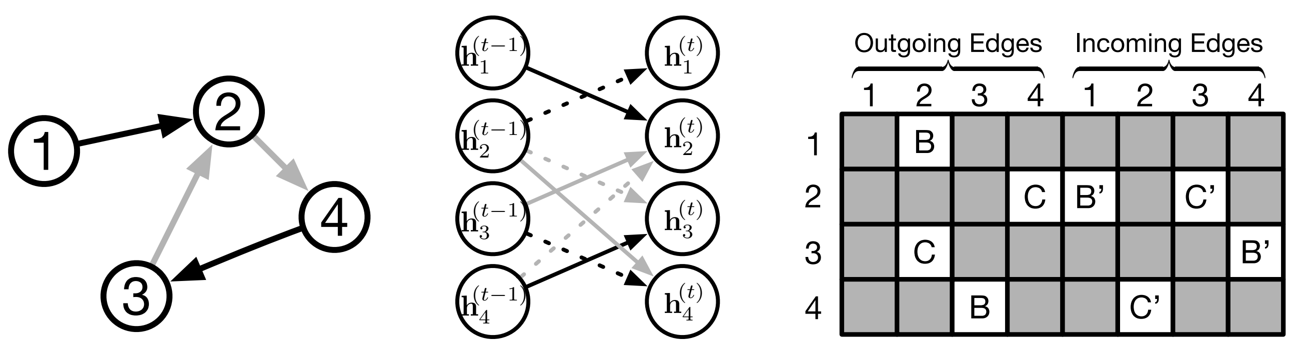
- Multi-agent Reinforcement Learning
  - Lots of papers on collaborative task solving
  - But usually communication protocol fixed

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



Related Work (II)

- Graph Neural Networks
  - Gori et al., IJCNN 2005;
  - Scarselli et al., IEEE Trans. Neural Networks, 2009
- Gated Graph Neural Networks
  - Li, Zemel, Brockschmidt & Tarlow, ICLR 2016.



CommNet 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

- **Learning Multiagent Communication with Backpropagation,** Sainbayar Sukhbaatar, Arthur Szlam, Rob Fergus, NIPS 2016
- Code: https://github.com/facebookresearch/CommNet

Learning Abstraction with NN (ongoing work)

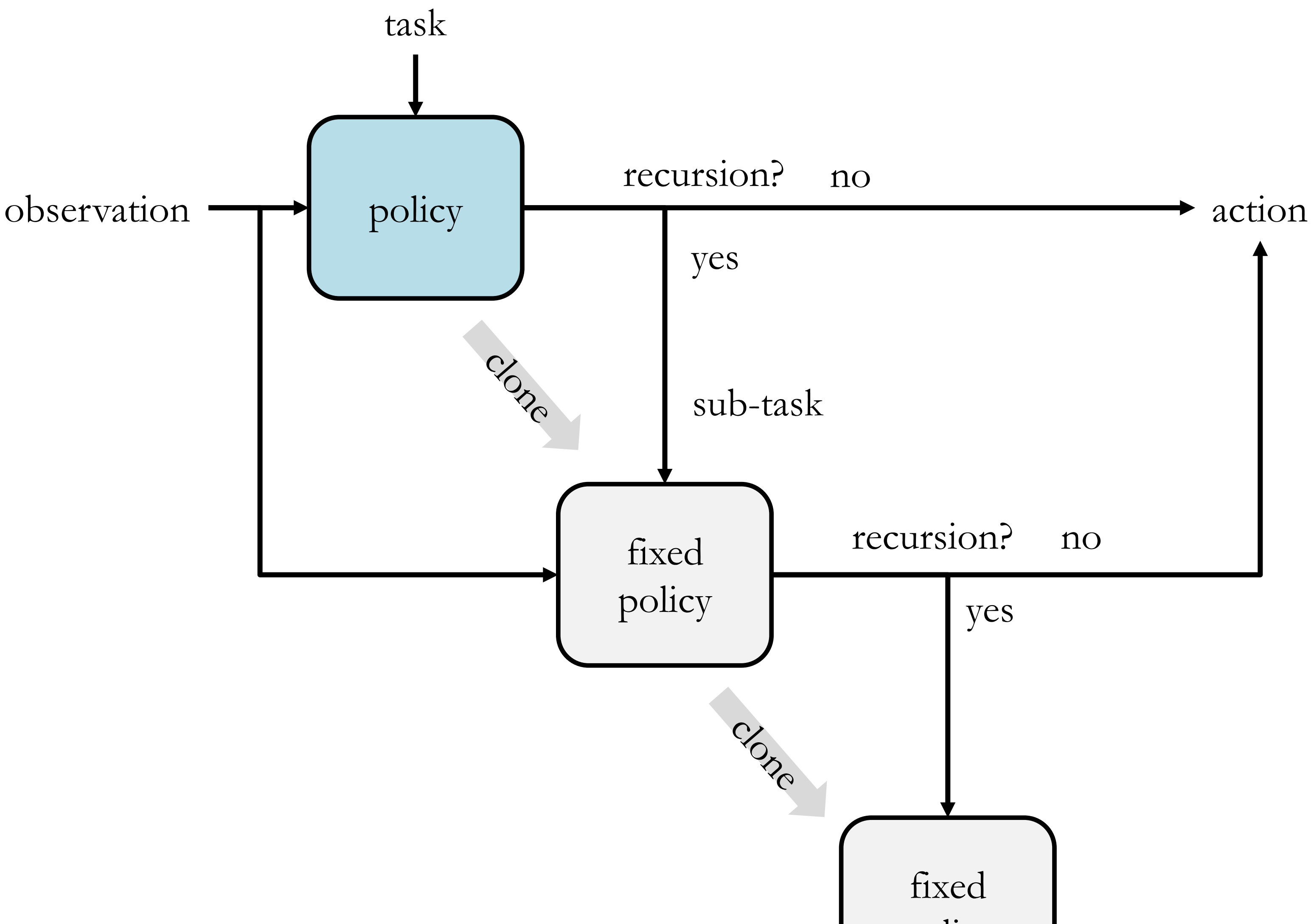
- Abstraction is vital in cognitive tasks
- Abstraction of observations
  - higher layers of ConvNets, hidden state of RNNs
- Abstraction of actions in reinforcement learning

- Going to work = thousands of steps/actions = get to station → take a train → walk to office
  - Deep RL models lack abstraction/hierarchy
  - Possible solution: recursive policy

Task-conditional RL

- Usual RL settings have fixed task
  - Policy(Observation) → action
- Key idea: Generalize so agents also perceive task
  - Policy(Observation, **Task**) → action
- Different tasks for different episodes
- Tasks can be represented by embedding vector

Recursive Policy



Execution of recursive policy

```
function Run(s, g, t)
    while t < T_max do
        a' ← π(s, g)
        if a' ∈ A then
            s ← Environment.Act(a')
            t ← t + 1
        else if a' = <term> then
            return s, t
        else
            g' ← a'
            s, t ← Run(s, g', t)
        end if
    end while
end function
```

- Task Ids: $g \in G = \{1, ..., K\}$
- Environment actions: $a \in A = \{1, ..., M\}$
- Extended actions: $a' \in A' = A \cap G \cap \{<\text{term}>\}$
- Policy: $\pi : s \times g \rightarrow a'$

- During training, give small internal reward for using recursion
- Size of reward slowly increases with epoch

Related Work

- **Options framework** [Sutton'99], **HAMs** [Parr'98], **MAXQ** [Dietterich'00]

- P. Bacon, J. Harb, D. Precup. "**The Option-Critic Architecture**", 2016.

- J. Oh, S. Singh, H. Lee, P. Kolhi. **"Communicating Hierarchical Neural Controllers for Learning Zero-shot Task Generalization"**, ICLR Submission, 2016

- J. Andreas, D. Klein, S. Levine. "**Modular Multitask Reinforcement Learning with Policy Sketches"**, ICLR Submission, 2016 (*borrow their tasks*)

- J. Cai, R. Shin, D. Song. **"Making Neural Programming Architectures Generalize via Recursion",** ICLR Submission, 2016

Representation of tasks

- Fixed number of tasks → use their unique IDs
  - **1**=“go to door”, **2**=“lock door”, **3**=“close door”
- Future: use natural language to describe a task
  - Task = list of words
  - Ex) “close door”, “close all doors”, “find open door”
  - Generalization to unseen tasks?

Random Difficulty During Training :

walk to X

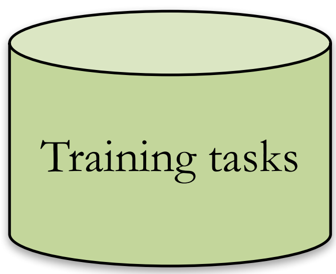
stand

grasp X

make coffee

bring X

put X on Y



Random Difficulty During Training :

Learns easy tasks
without recursion

Learn to solve hard tasks by
recursively calling easy tasks

time

stand

grasp X

walk to X

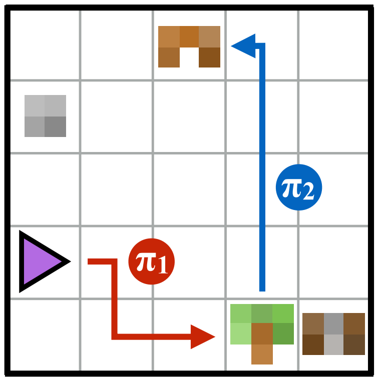
bring X

put X on Y

make coffee

Training tasks

Preliminary results



[Andreas et al. '16]

Tasks are randomly sampled from:

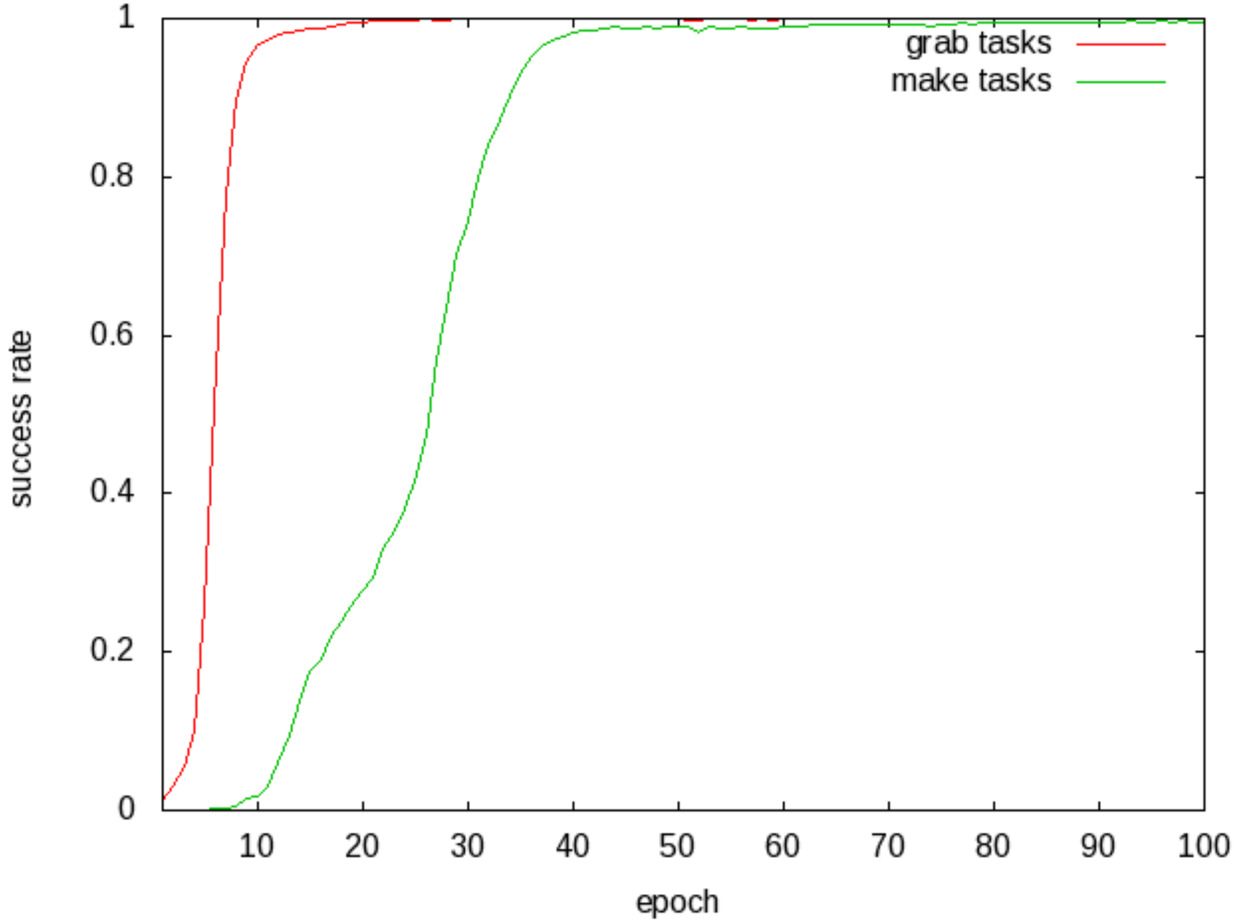
1. Grab wood
2. Grab iron
3. Grab rock
4. Make axe (wood, iron → worktable)
5. Make sword (rock, iron → factory)
6. Make hammer (wood, rock → toolshed)
7. Make bridge(wood, iron, rock → plant)

Model doesn't know which ones are easy
No specification of sub-tasks in Make tasks

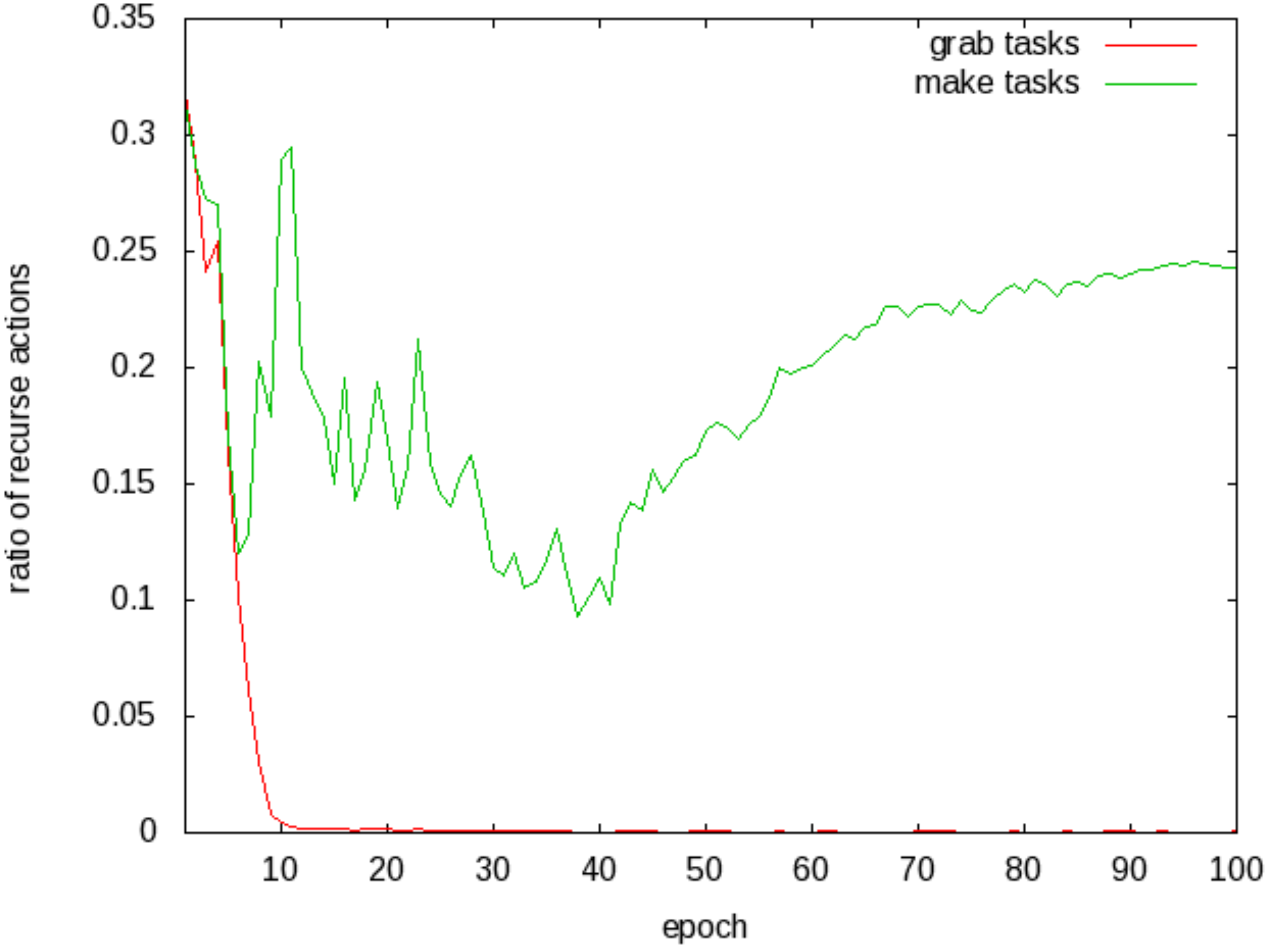
Training Details

- Using MazeBase [Sukhbaatar et al.'16]
- Policy is fully-connected NN with 2 hidden layers (50 units/layer)
- Trained with REINFORCE [Williams'92]
- Reward structure:
  - Each time step: -0.1
  - Complete task: +1
  - Use recursion (1 level): +0.02 [ramp; initially zero]

Model learns easy tasks first

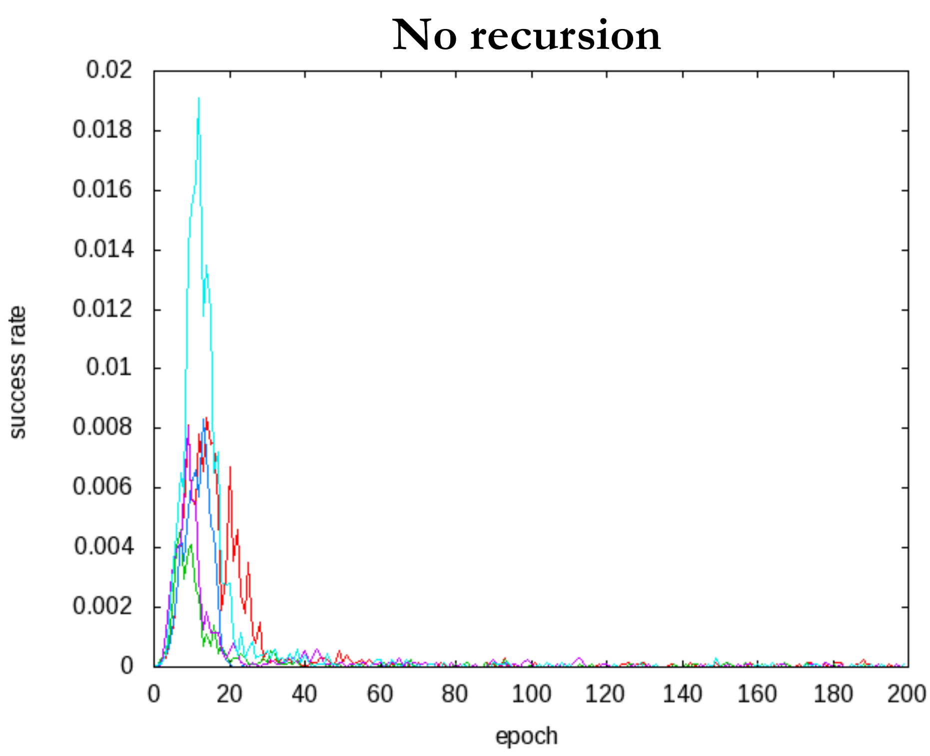


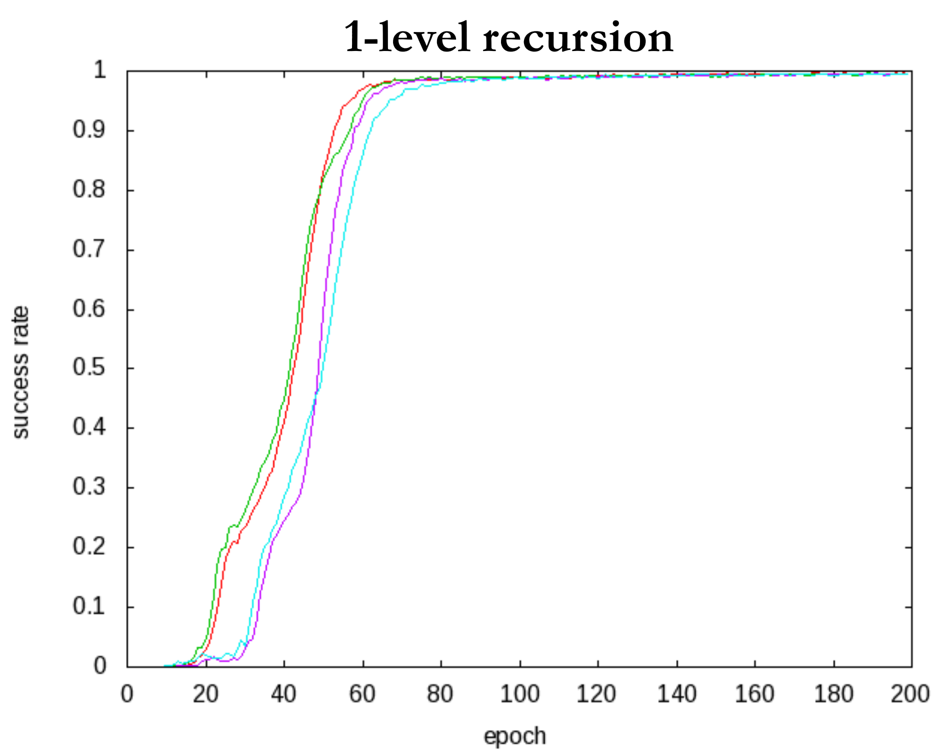
Model learns when to use recursion



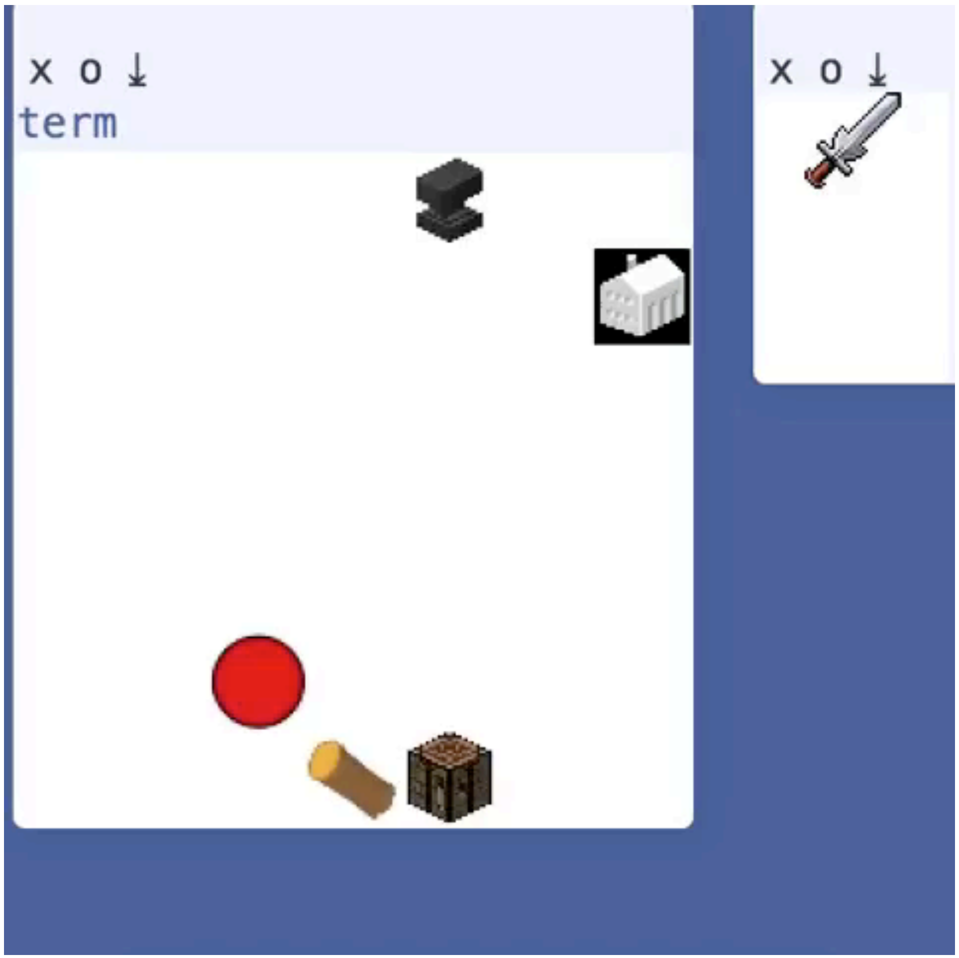
Comparison with non-recursive model

- Success rates on harder "Make" tasks
- 20 hidden units/layer

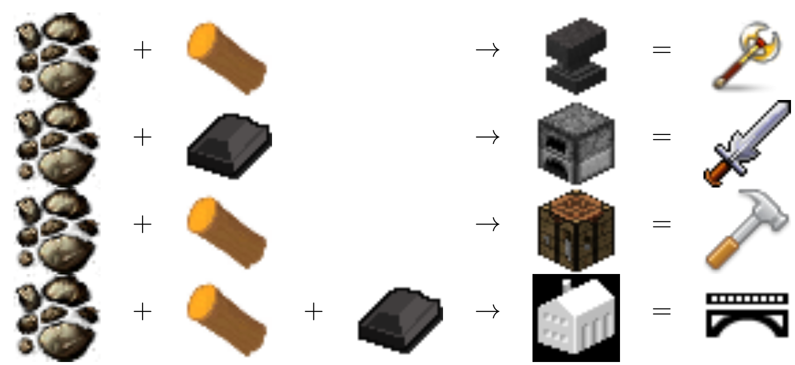




Trained Model



Task key:



Summary

- Simple recursion-based RL approach
- Learns sub-task structure with minimal supervision

Future work:

- More than two levels of hierarchy
- More complex environment with diverse tasks
- Natural language for task description
- Learn to create a novel sub-task

Experiment: bag to sequence

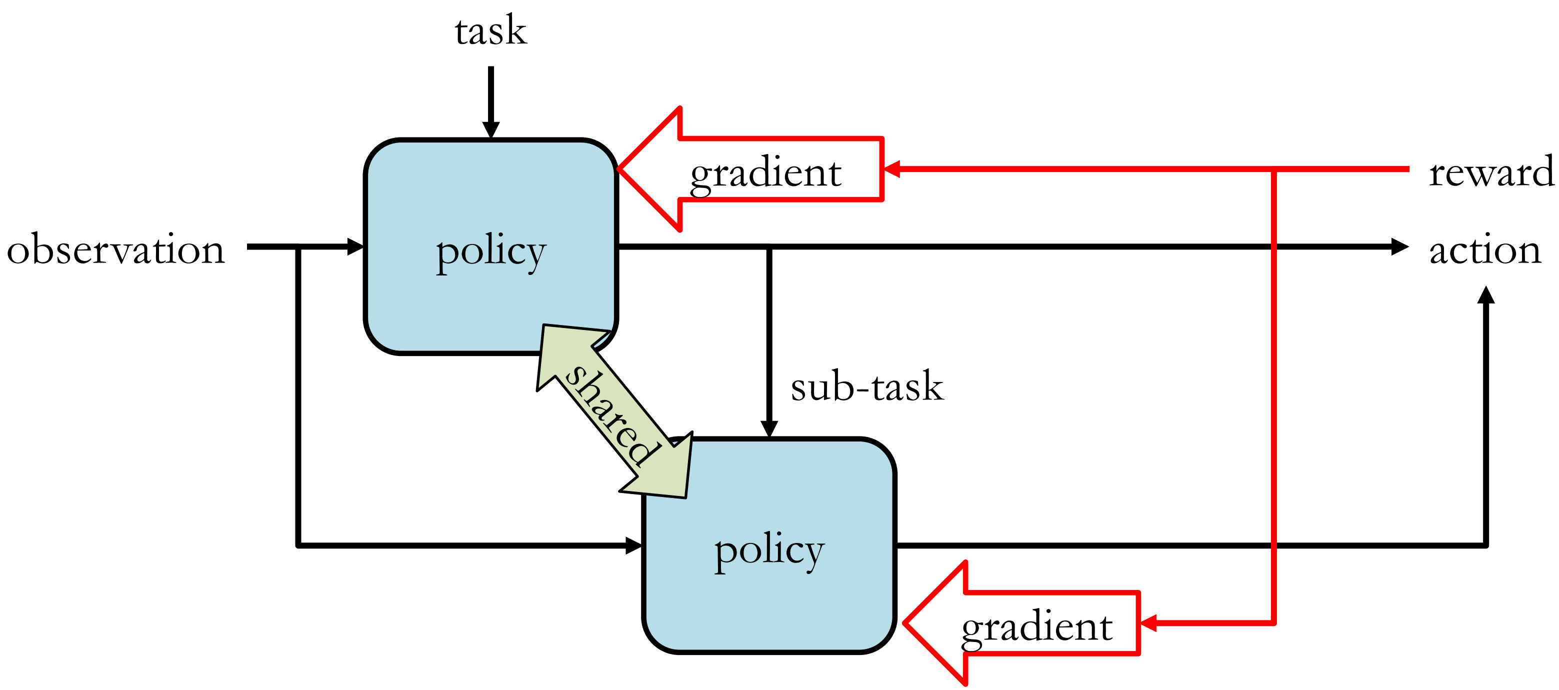
- Problem: given a set of words, arrange them in right order.

{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%

Fine-tuning of lower policies



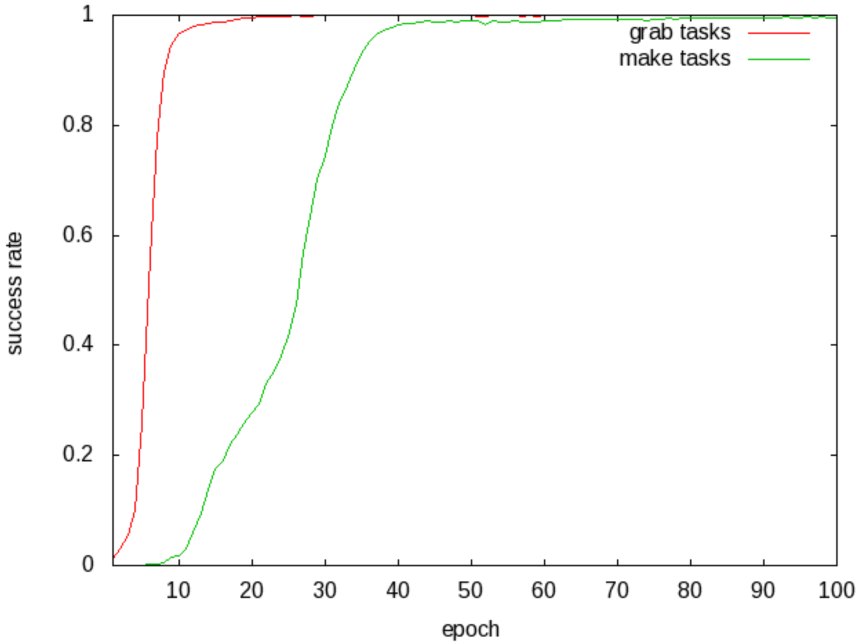
The whole model a stochastic computation graph → All discrete actions including internal decisions can be trained with policy gradient.

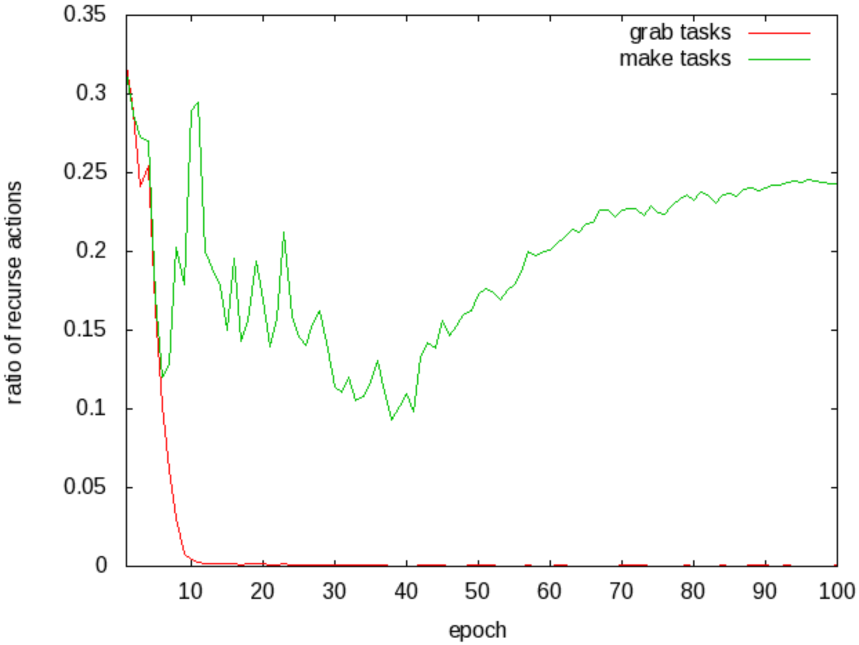
Only fine-tune lower policies, and not change its behavior

Initial experiments in grid world

	Failure rate on visit two goals	Reward + recursion reward
No recursion	3.11%	0.213
Recursive	1.02%	0.501
Recursive + fine-tune	0.11%	0.561

Model learns easy tasks first

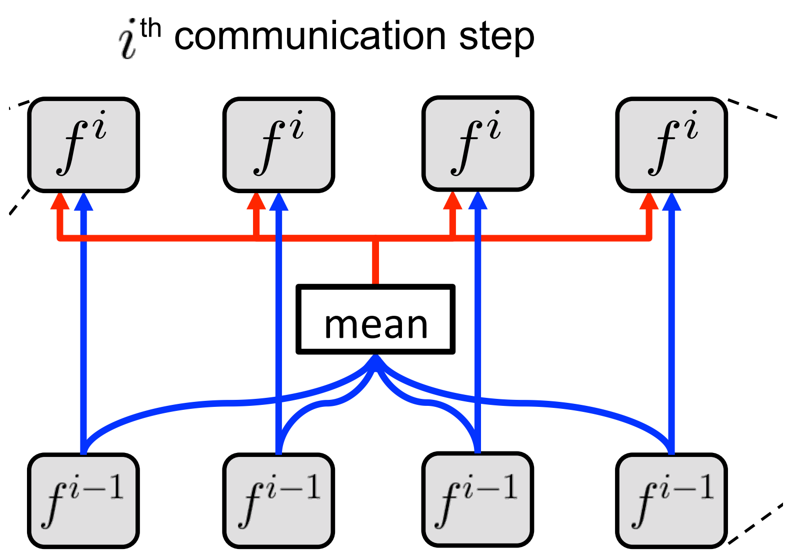




Not using recursion when task is easy

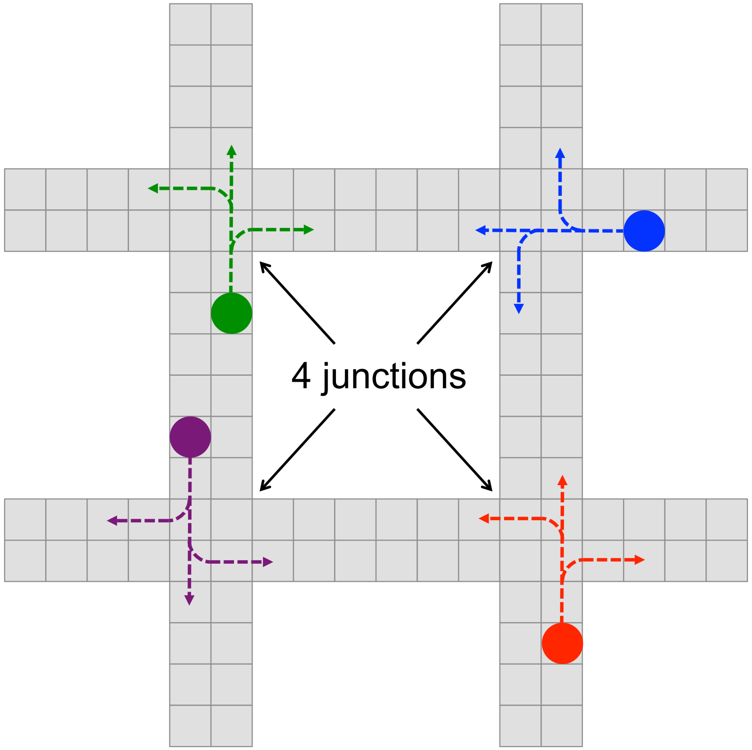
Communication Structure

- Broadcast channel between all agents
- Each agent receives average of hidden state from all other agents: $c_j^i = \frac{1}{J-1}\sum_{j' \neq j} h_{j'}^i$



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



Lever pulling task

- 500 agents with unique IDs
- In each episode, 5 of them randomly chosen
- There 5 different levers to pull
- If all agents pull different levers → WIN

Communication	Training method: Supervised	Training method: Reinforcement
None	0.59	0.59
CommNet	0.99	0.94

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%