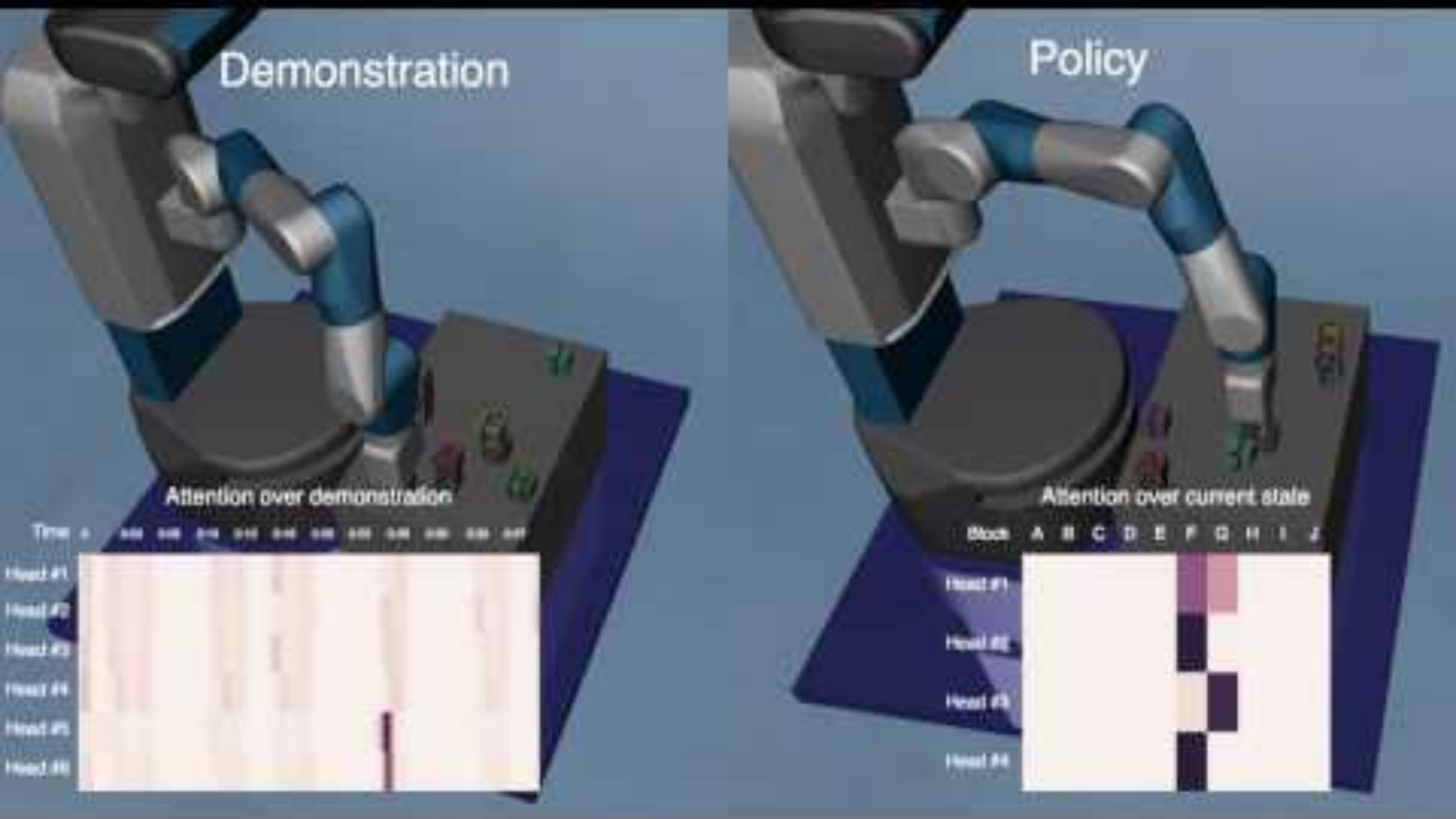
One-Shot Imitation Learning

Yan Duan, Marcin Andrychowicz, Bradly Stadie, Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, Wojciech Zaremba

Motivation & Problem

- Imitation Learning commonly applied to isolated tasks
- Desire: Learn from few demonstrations; instantly generalize to new situations of same task
- Consider the case where there are infinite tasks, each with various instantiations (initial states)



Demonstration
Policy
Attention over demonstration
Time
Head #1
Head #2
Head #3
Head #4
Head #5
Head #6
Attention over current state
Block
A
B
C
D
E
F
G
H
I
J
Head #1
Head #2
Head #3
Head #4

Method Overview

Train

-) Demonstration
-) State

→

-) "Optimal" action for State

Test

-) Demonstration
-) State

→

-) Action

Architecture

3 Neural Networks

- Demonstration Network
- Context Network
- Manipulation Network

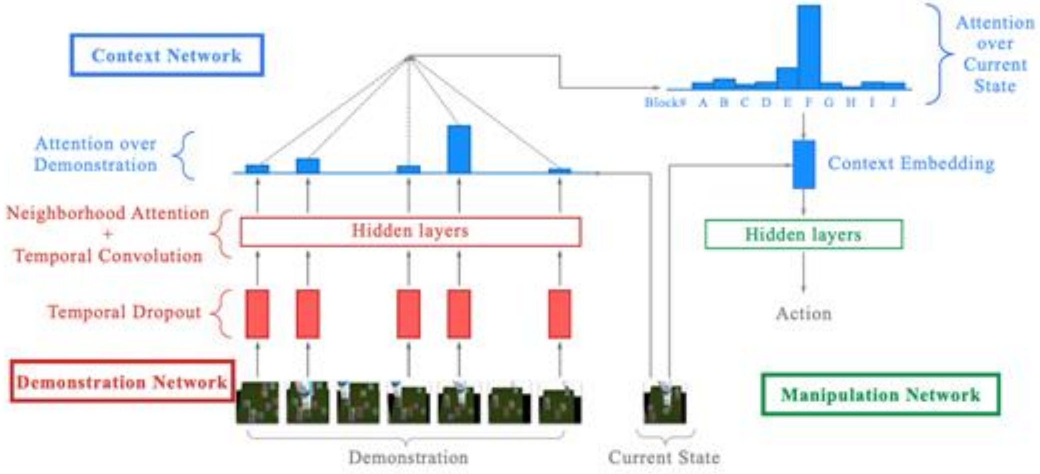


Figure 2: Illustration of the network architecture.

Demonstration Network

- Receives a demonstration trajectory (seq of frames) as input
- Produces an embedding of the demonstration to be used by the policy
- Embedding grows linearly w/ length of demonstration & number of blocks
- Temporal dropout (throw away 95% of training timesteps) for tractability
- Dilated Temporal Convolution (capture info across timesteps)
- Neighborhood Attention: maps variable-dimensional inputs to outputs with comparable dimensions.
- Thus, unlike soft attention, (single output), we have as many outputs as inputs, where each output attends to all other inputs in relation to its own input.

Context Network

- Input: 1) current state and 2) embedding produced by the demonstration network
- Output: a context embedding, independent of length of demonstration and number of blocks
- Temporal attention over demonstration embedding: produces a vector whose size is proportional to the number of blocks in the environment.
- Attention over current state: produces fixed-dimensional vectors, where memory content consists of positions of each block, which, concatenated to the robot’s state, forms the context embedding.
- **Key intuition:** Number of relevant objects usually small and fixed. Eg, source and target block. Need fixed dimensions, unlike demonstration embedding.

Manipulation Network

- Computes the action needed to complete the current stage of stacking one block on top of another one
- Simple MLP network
- Input: Context Embedding
- Output: N-dimensional output vector for robot arm
- Modular training: doesn’t need to know about demonstrations or more than two blocks present in the environment (* open to further work)

Architecture

3 Neural Networks

- Demonstration Network
- Context Network
- Manipulation Network

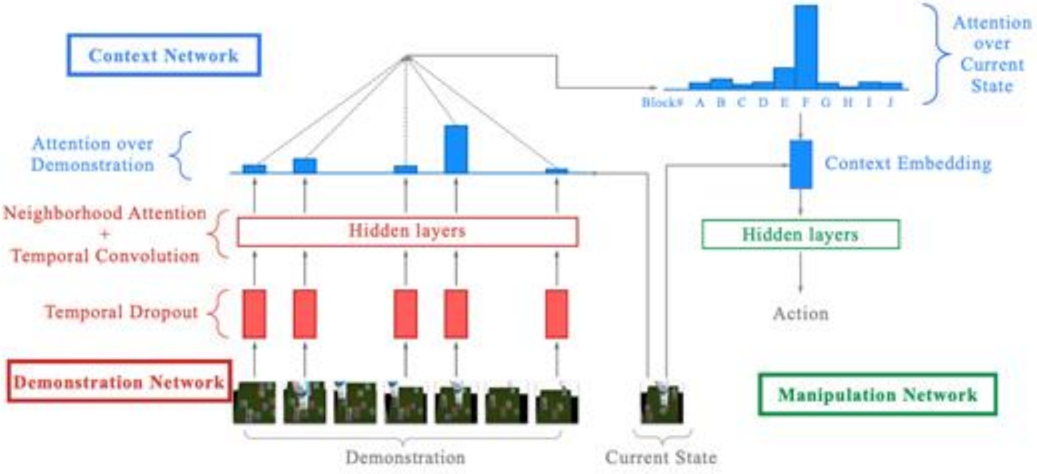


Figure 2: Illustration of the network architecture.

Brief Discussion

- Do you agree that stacking blocks on top of each other is a Meta Learning Problem?
- What kinds of other tasks could this problem setup generalize to, if successful?

Experiments

Key questions to investigate/answer:

1. Comparing training schemes: behavioral cloning vs. DAGGER
2. Effect of conditioning on different slices of data
    i. Entire demonstration (original method)
    ii. Final state
    iii. Snapshots of trajectory (hand-selected informative subset of frames)
3. Generalizability of the framework

- Behavioral cloning: directly learn policy using supervised learning
- DAGGER (Ross, Gordon, and Bagnell 2011) : repeatedly aggregate data by labeling paths taken by learned policy and adding them to data

Experiments

Setup

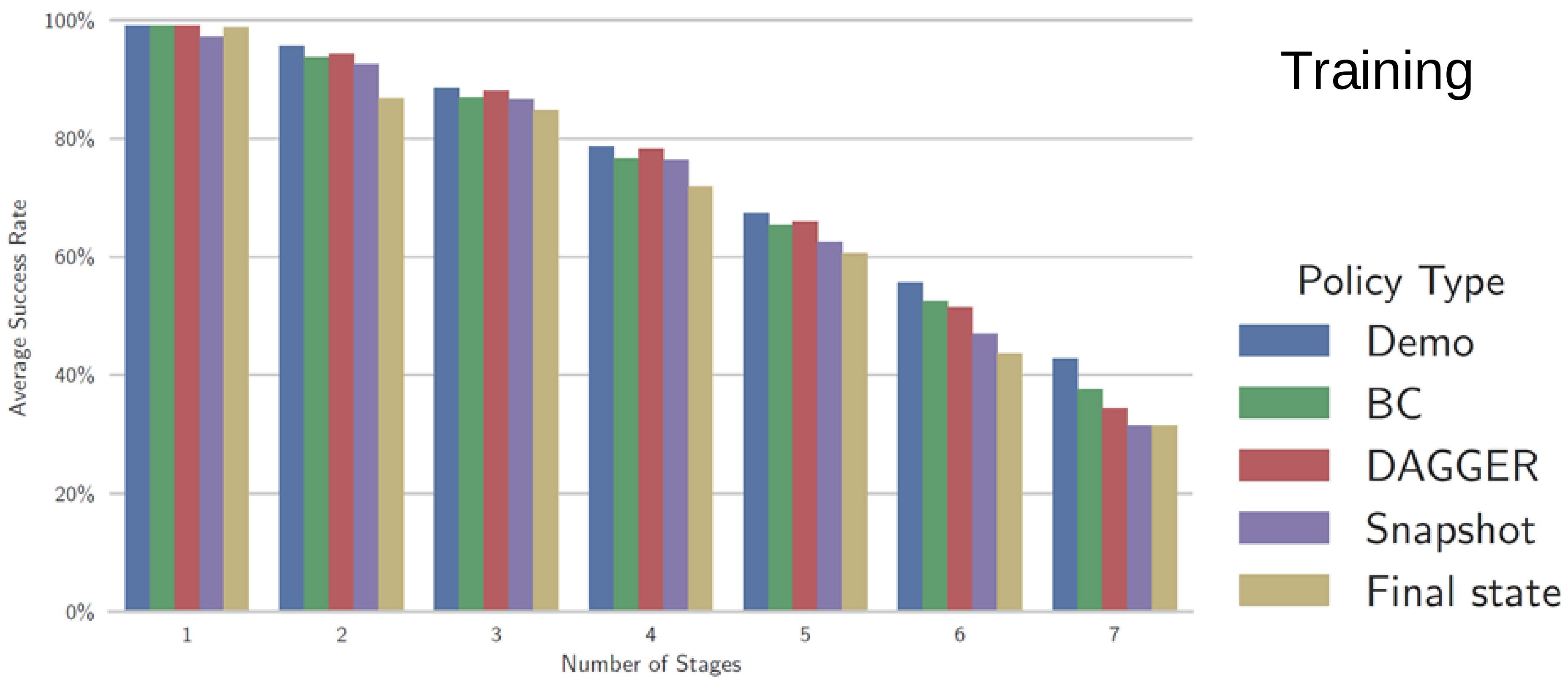
- 140 training, 43 test tasks; each with 2 ~ 10 blocks with different layouts
- Collect 1000 trajectories per task using hard-coded policy

Models compared

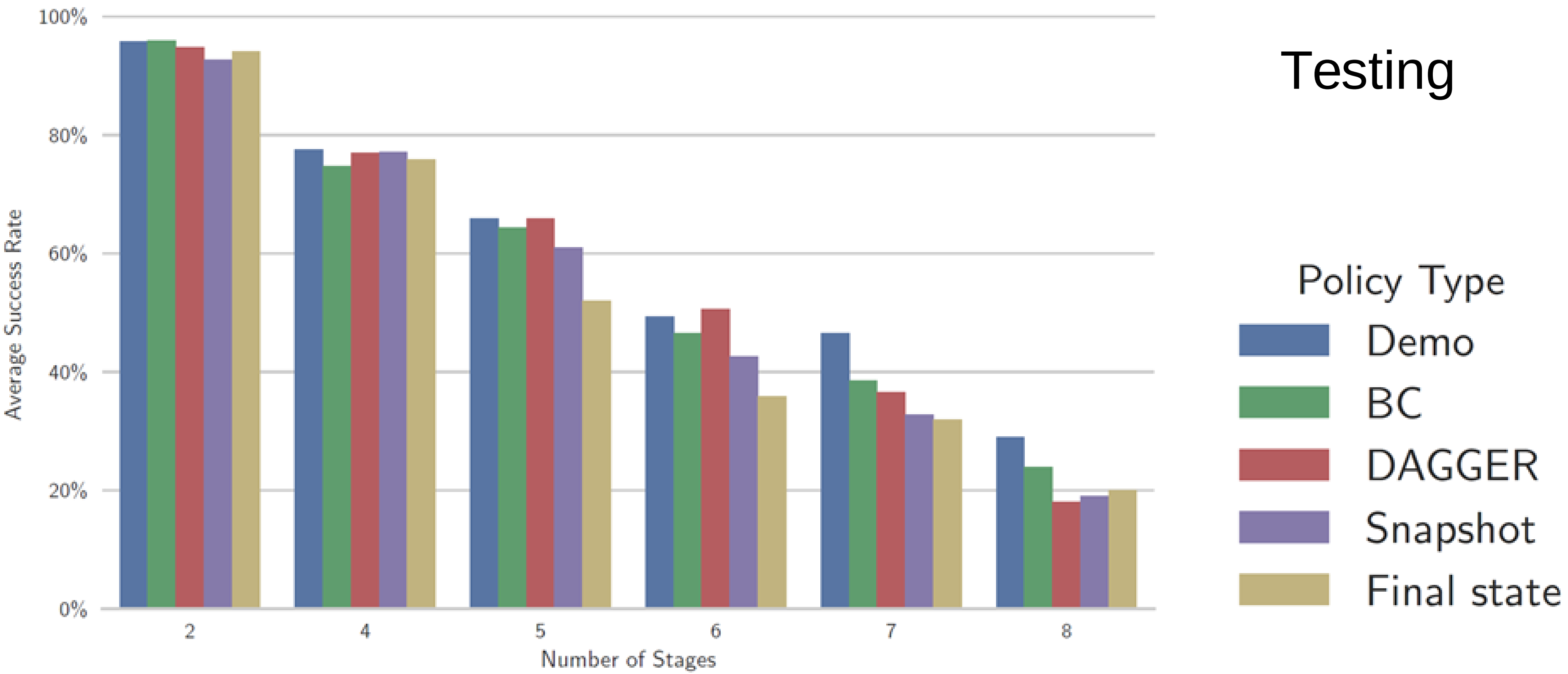
1. Same architecture, trained with behavioral cloning
2. Same architecture, trained with DAGGER
3. Conditioning on final state, trained with DAGGER
4. Conditioning on snapshots (last frames of each “step”), trained with DAGGER

How do you expect them to perform?

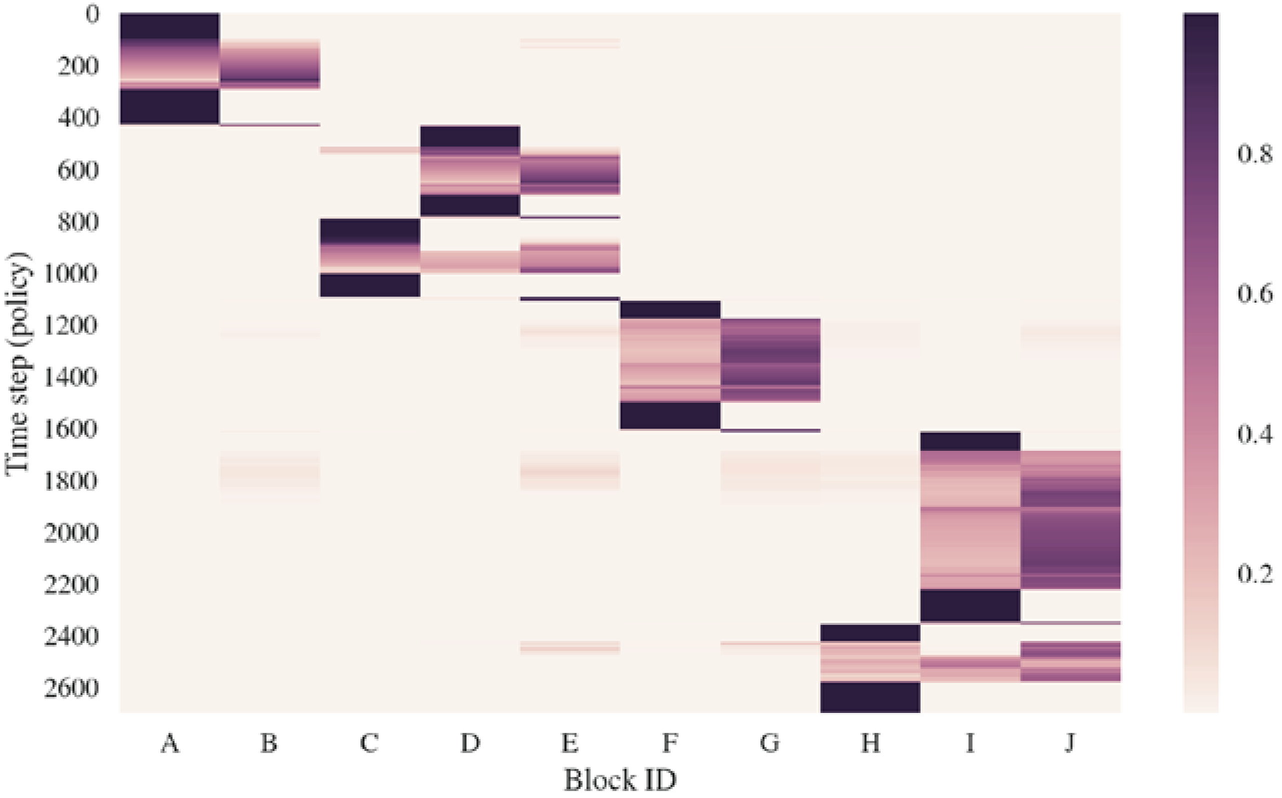
Experiments



Experiments



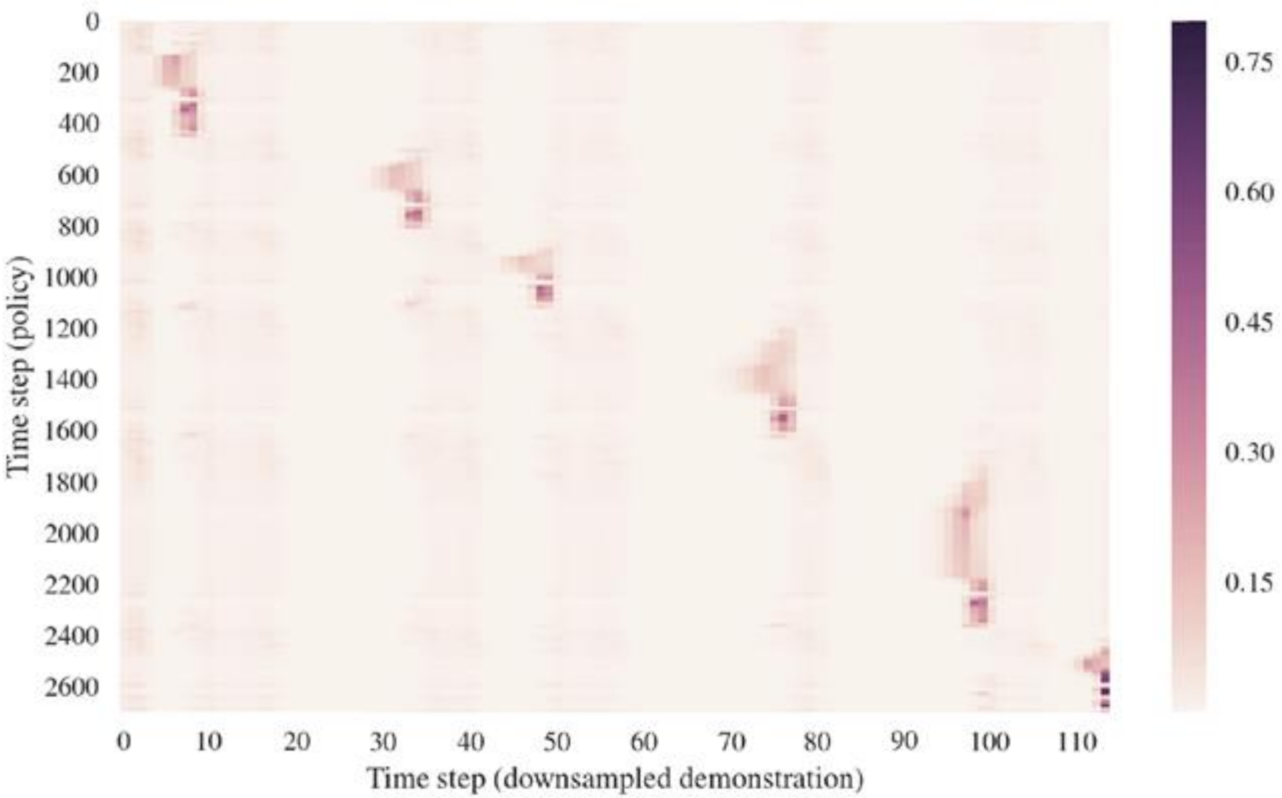
Experiments



Attention over blocks

Configuration:
ab cde fg hij

Experiments



Attention over time steps

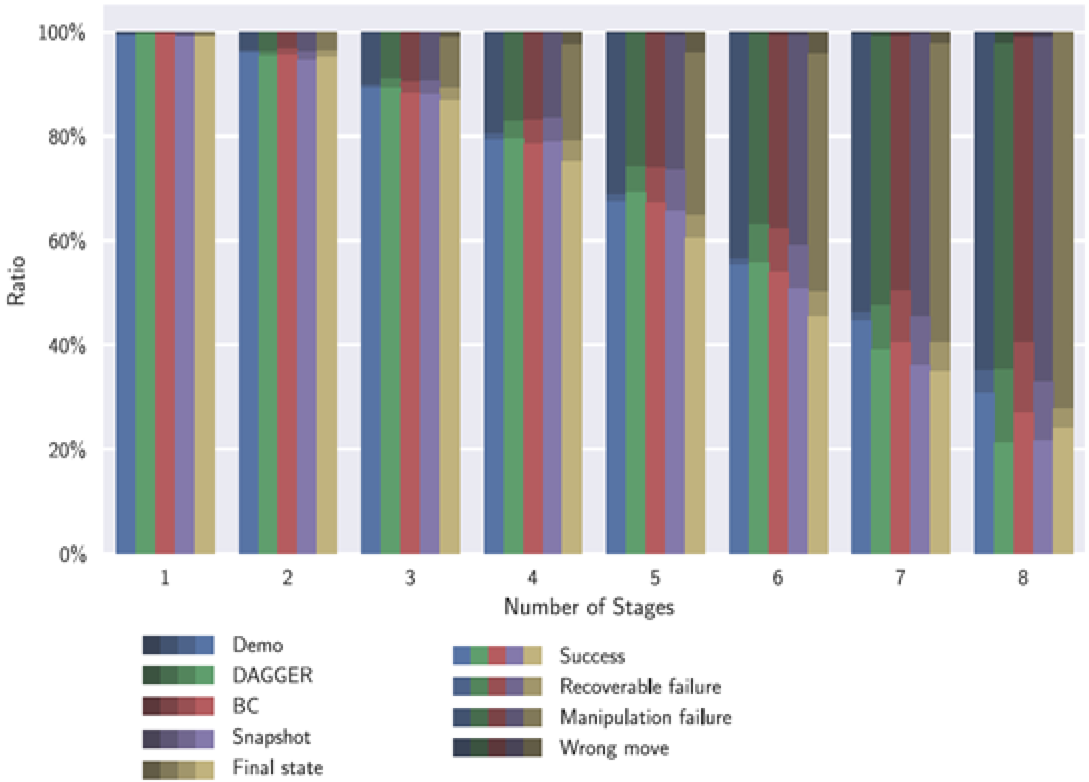
Configuration:
ab cde fg hij

Experiments

Breakdown of failures

- **Wrong move**: layer incompatible with desired layout
- **Manipulation failure**: irrecoverable failure
- **Recoverable failure**: runs out of time before finishing task

A lot of manipulation failures



Takeaways / Strengths

- Learning a family of skills makes learning/performing relevant tasks easier
- Interesting breakdown into **modular structure**
- Some results are very **intuitive** and clear, as exemplified by attention
- **Neighborhood attention** maps inputs of variable size to comparable dimension outputs and extract relationship between itself and others
- **Single-shot** learning result is rather impressive
- While not presented in this paper, the data was collected using **simulations** rather than actual images (vision system never trained on real image)

Weaknesses / Limitations

- Performance depends on manual collection of “optimal” demonstrations.
- The tasks are all very similar - stacking blocks into 1 tower is very similar to stacking blocks into 2 towers. How much generalization is really happening ?
- Algorithm immediately fails on unrecoverable state - no best effort to finish. Ex, when a block falls off the table.
- Authors assume that the distribution of tasks is given, and that they can obtain successful demonstrations of each task. How often is this true?
- It is rather tough to comprehend the structure of the network without taking a close look at the algorithm in the appendix.
- Single experiment task discussed - they mention another task in appendix, but is very simple, and does not use architecture in paper. Can the network be utilized for other tasks?
- Action space is never really defined/explained throughout the paper

Further questions

- Could the model learn to “disassemble” the blocks?
- Can the starting position be stacked?
- To what degree can the model correct its mistakes?
- How do “number of moves” or time compare across algorithms?
- Were the attention plots carefully selected? Or do they portray the behavior in general.
- How does model perform if we selected “random” snapshots?
- How much ‘noise’ can demonstration include?

Discussion Questions

- What applications could this be useful for?
- How would we condition on multiple demonstrations, rather than a single one?
- On a similar note, can we supply “feedback”, as a teacher to a student would do? (Something like DAGGER, but test time?)

Appendix