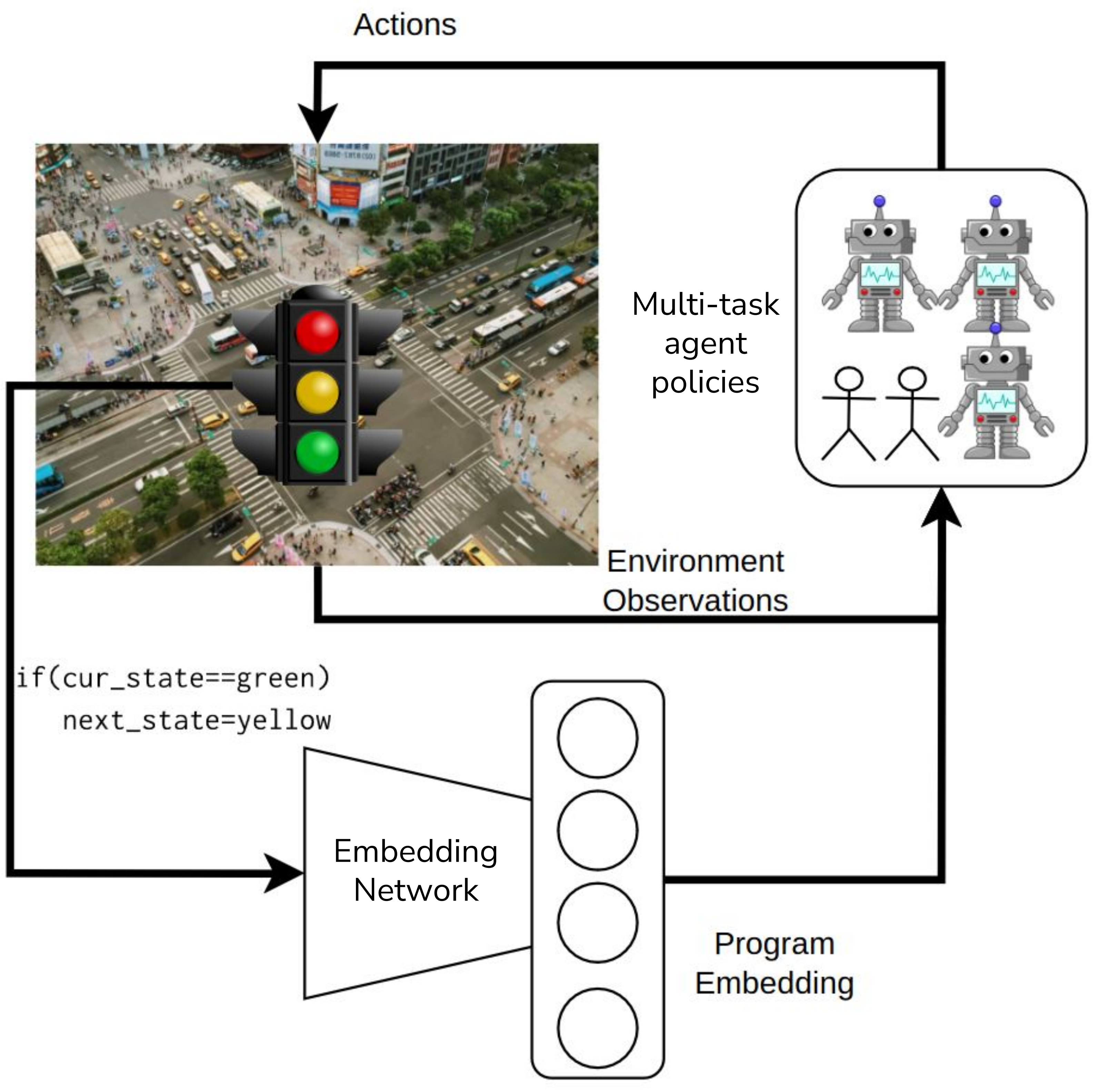
Program Embeddings for Rapid Mechanism Evaluation

Sai Kiran Narayanaswami[1], David Fridovich-Keil[1], Swarat Chaudhuri[1], Peter Stone[1,2]

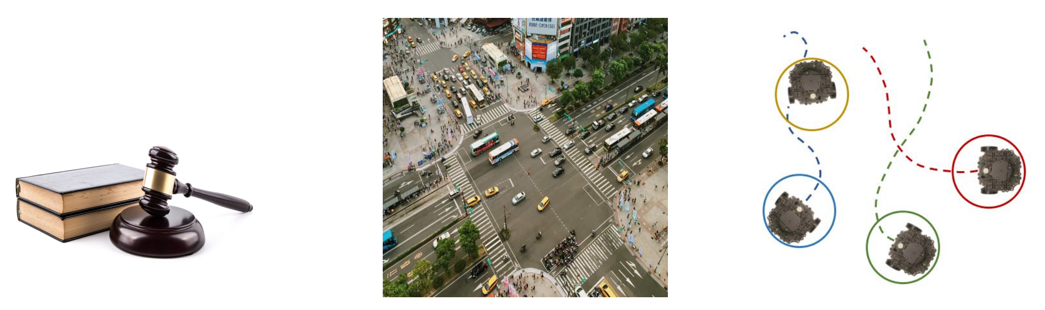
[1]The University of Texas at Austin [2]Sony AI

I. Rapid mechanism evaluation can enable the design of mechanisms in the form of programs.
II. Task-conditioned learning efficiently learns agent strategies for a large set of mechanisms.
III. Program embeddings serve as informative task context for inferring equilibrium strategies.



Mechanism Design

Configure multiagent systems (mechanisms) to maximize a global outcome when agents act in their own interests.



Program Synthesis for Mechanism Design

Mechanisms are represented as programs in a Domain Specific Language (DSL).

Programmatic Mechanism Design: Find a program from a search space that maximizes a global outcome for given assumptions about agent behavior (e.g Nash solver, Multiagent RL algorithm).

Benefits: programs are interpretable, verifiable.

Challenges and Motivations

- Program search is hard: large, combinatorial search spaces.
- Program synthesis algorithms require iterative evaluation of candidate programs (mechanisms).
- Mechanism evaluation requires knowledge of resulting agent behavior: finding Nash equilibria can be intractable, learning algorithms are expensive.

How can we quickly evaluate mechanisms from large, programmatic search spaces?

Solution: Embeddings as Task Context

Multi-Task Learning:

Efficiently learn one set of agent strategies to solve multiple tasks (mechanisms) based on task context.

What is a good task context? Cannot treat each program in the search space as a separate task: too many programs.

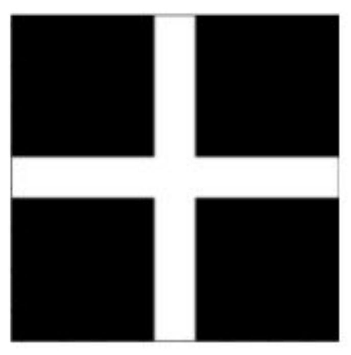
Program Embeddings: Convert high-dimensional program source code into a lower-dimensional vector space.

Rapid Mechanism Evaluation Framework

- Train agent policies using any multi-task learning algorithm with program embedding as task context.
- No learning needed to evaluate a new mechanism: simply use its embedding to run the agent policies
- Embedding network and agent policies can be reused for different mechanism design objectives.

Experiments

- 2-player, zero-sum matrix games with 0/1 entries.
- Payoff matrices generated by simple programs with “set” statements.
- Pre-trained Code2Vec[2] embedding network (code vectors).

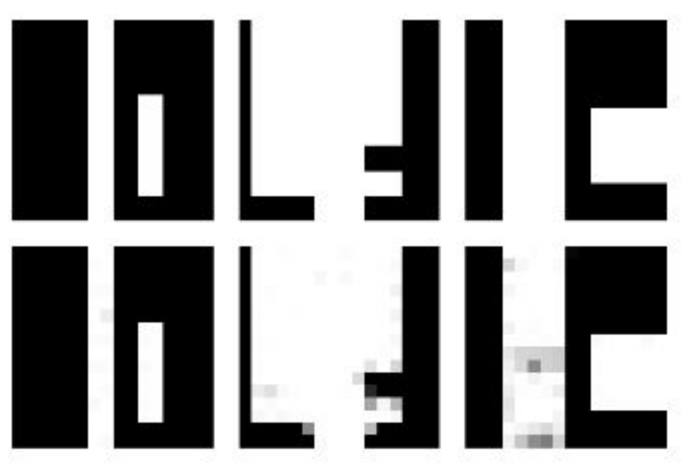


```
set(7, 9, 0, 15, 1);
set(0, 15, 7, 9, 1);
```

Matrix Reconstruction

Reconstruct the game matrix using only the code vector.

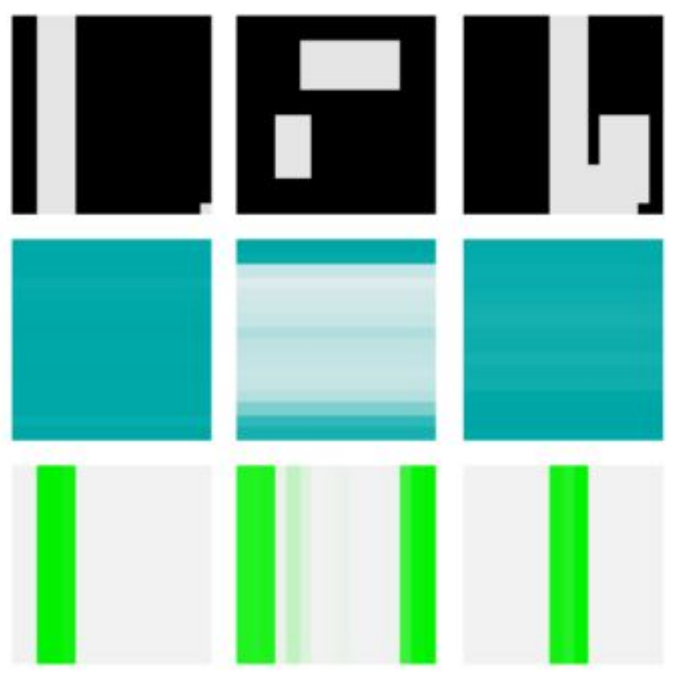
Trained to minimize MSE loss between predicted and ground-truth matrices.



Nash Strategy Prediction

Train policies that closely mimic the Nash strategies for each agent.

Use Behavioral Cloning to imitate Nash strategies (PATH solver[3, 4]) by minimizing KL divergence.



Conclusions and Future Work

- Off-the-shelf, general purpose Code2Vec embeddings are informative enough to infer equilibrium behavior.
- What happens in more complex, sequential environments with RL for agent behaviors?
- Fine-tuning large embedding models for scalability.

[1] https://faculty.engineering.asu.edu/acs/research/resilient-collective-systems
[2] Alon et. al., Code2vec: Learning distributed representations of code. (POPL 2019)]
[3] http://pages.cs.wisc.edu/~ferris/path.html
[4] Forrest Laine. TensorGames.