Queen Mary
University of London

Local Forward Model Learning for GVGAI Games

Alexander Dockhorn and Simon Lucas

Conference on Games 2020

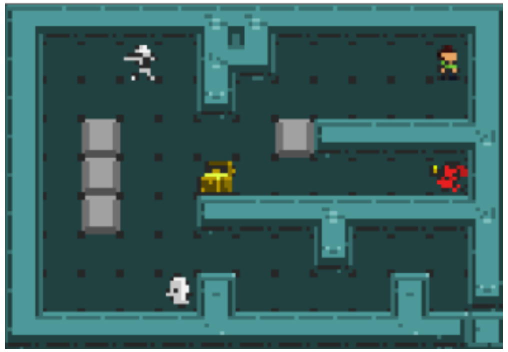
The GVGAI Learning Competition

- a competition by Hao Tong, Yang Tao and Jialin Liu -

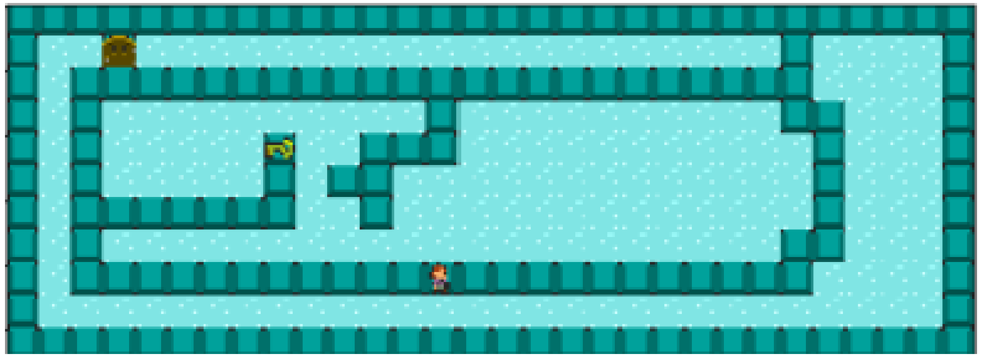
General Video Game AI (GVGAI) Learning Competition:

- train an agent on a set of levels of an unknown games
- play other levels of the same game without having them seen
- game-states are provided in a pixel-based state observation

Golddigger



Treasurekeeper



Waterpuzzle

Local Forward Model

What are Local Forward Models?

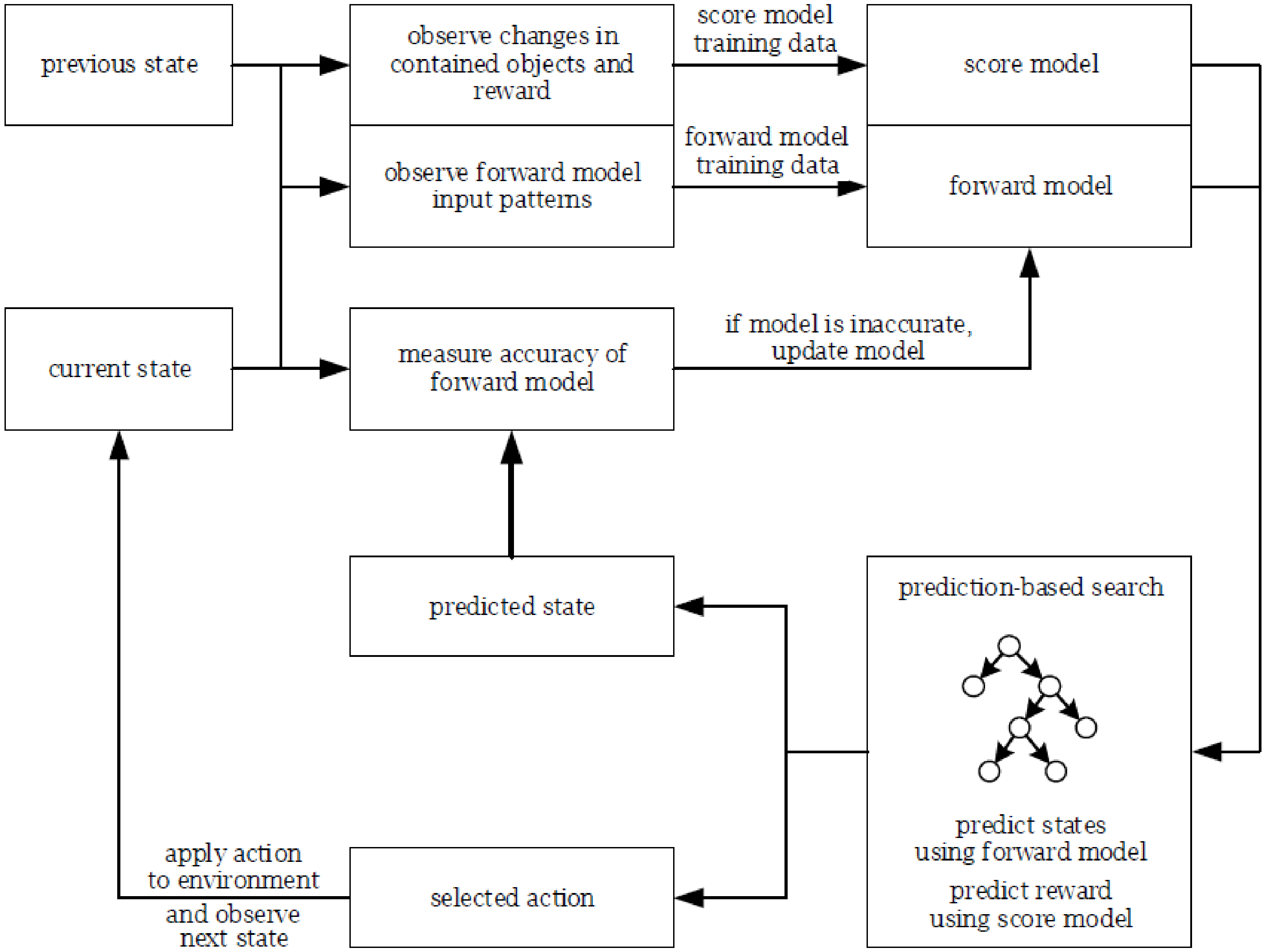
- Local forward models map represent a decomposed prediction of the next state
- The prediction of each component is only dependent on its current state and the state of surrounding elements

Why do we use them?

- Due the decomposition we can gather multiple training examples per time-step
- We want to study search-based methods in these game-learning scenarios

How do we use them?

Prediction-based Search



Modelling Local Dependencies

Assumptions:

- structured representation of the state
- requires a similarity or distance function for sensor values
- semantic of a sensor-value is independent of its index

Tile-based Representation (of Video Games):

- a state can be represented as a matrix T of size $n \times m$
- $T(x, y)$ specifies the observed tile at position (x, y)

Game-State of Sokoban

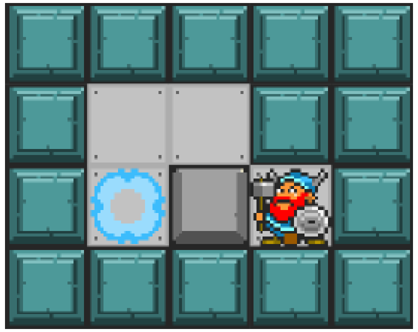
Tilemap Components

Local Transition Function

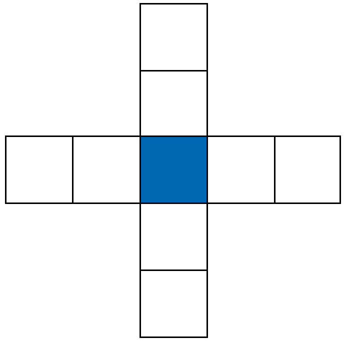
Decompose the forward model into one sub-model per tile:

$$\underset{x,y}{fm} : \left(N(x,y)_t,\ A_t\right) \longmapsto T(x,y)_{t+1}$$

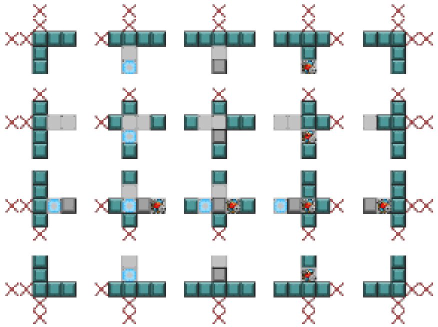
- N tile wi$(x, y)_t$ describes the local neighbourhood of tile $T(x, y)$ at time t
- it contains each th distance less than a given threshold



Game-State of Sokoban



Local Neighbourhood



Extracted Pattern(s)

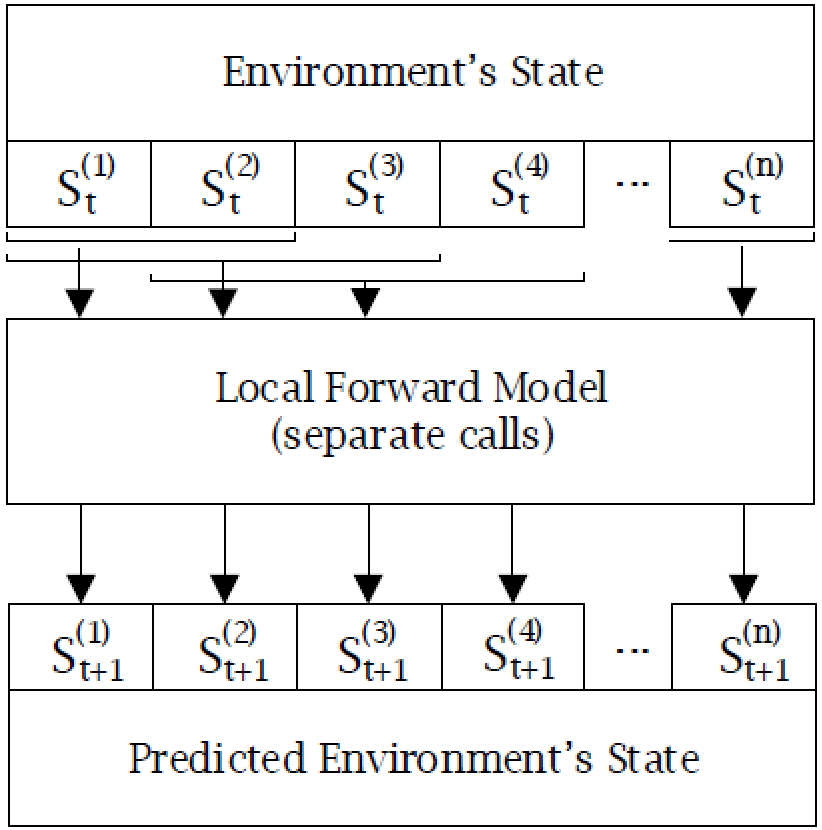
Local Transition Function

Local Forward Model

Predict the next state by predicting each tile

$$T_{t+1} = \begin{bmatrix} fm_{1,1}(N(1,1), A_t) & \dots & fm_{1,m}(N(1,m), A_t) \\ \vdots & \ddots & \vdots \\ fm_{n,1}(N(n,1), A_t) & \dots & fm_{n,m}(N(n,m), A_t) \end{bmatrix}$$

In case the semantic of a tile is independent of its position, only a single model needs to be learned



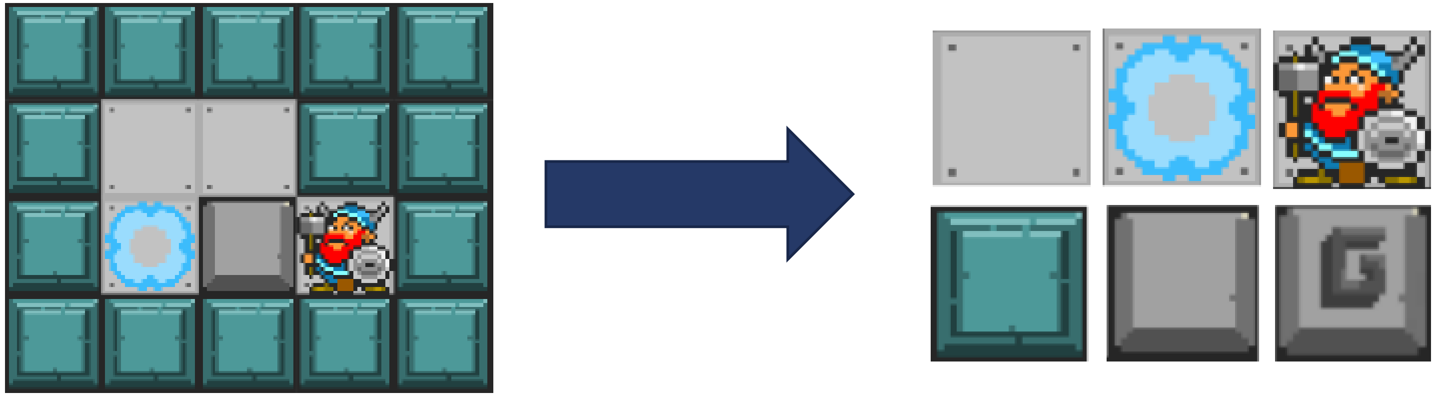
Advantage: higher sampling efficiency

- each observed state transition consists of one observed pattern per tile (in total: $n \times m$ patterns)

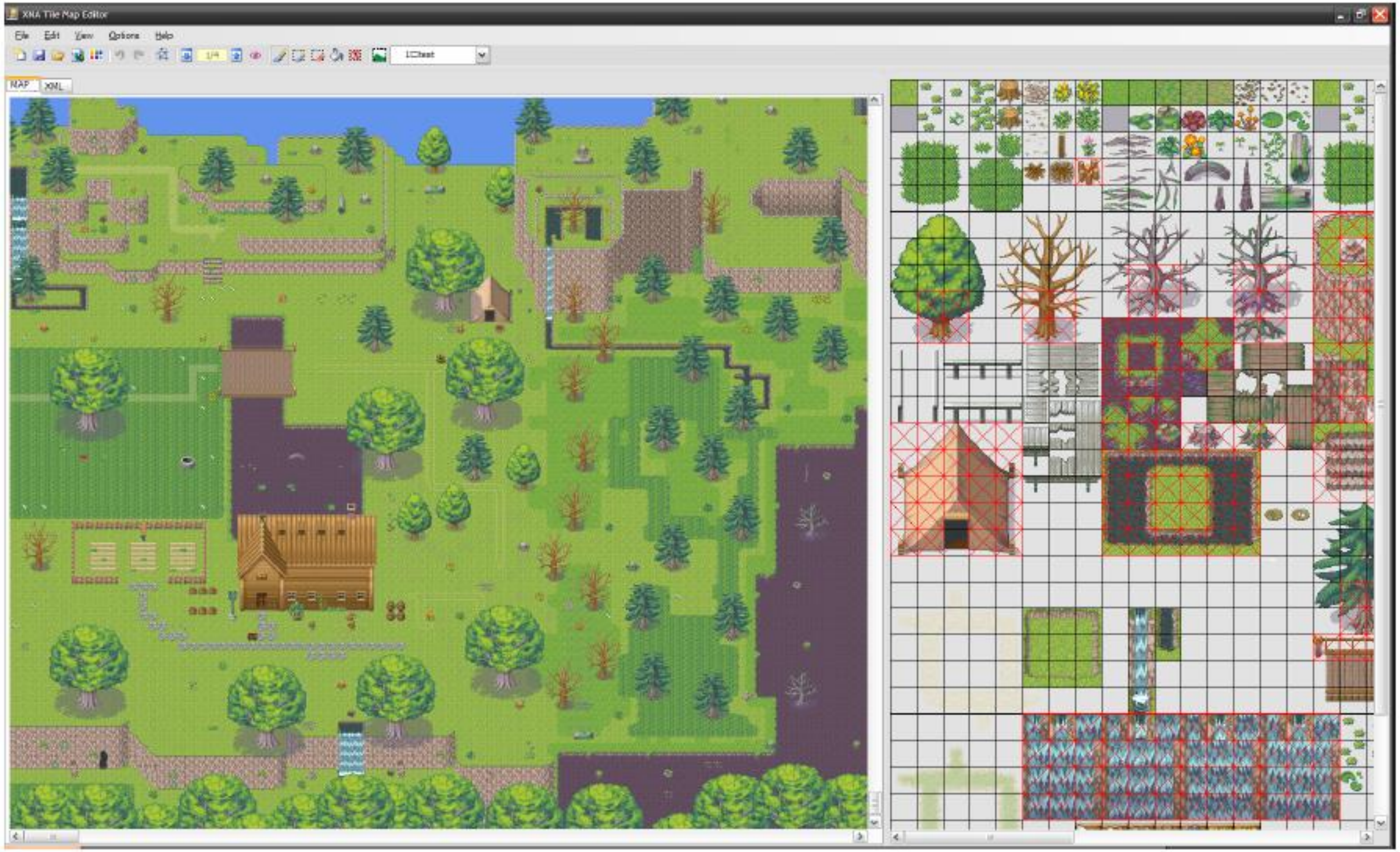
Pre-processing Pixel-based Input

Motivation:

- Local forward models can be applied to pixel-based state representations but require a large neighbourhood pattern
- Increasing the number of pixels to be considered exponentially increases the number of observable patterns
- Preprocessing the pixel-based state representation may improve the efficiency of the training process



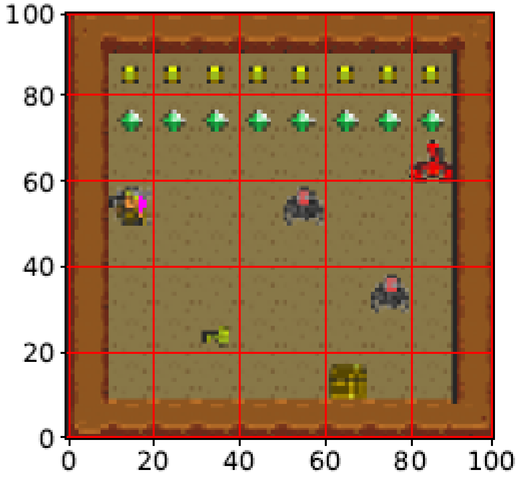
Tilemaps Example[1]

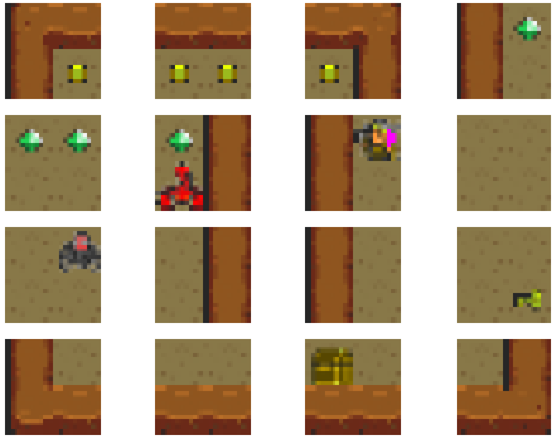


[1] https://xnafantasy.wordpress.com/2008/11/22/xna-map-editor-version-30-released/

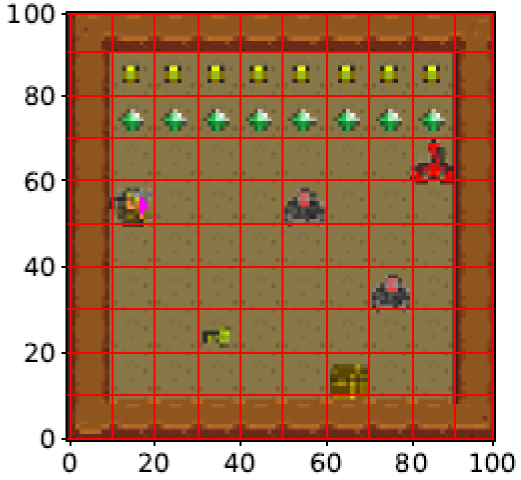
Pre-processing Pixel-based Input

Tile-size = 20





Tile-size = 10



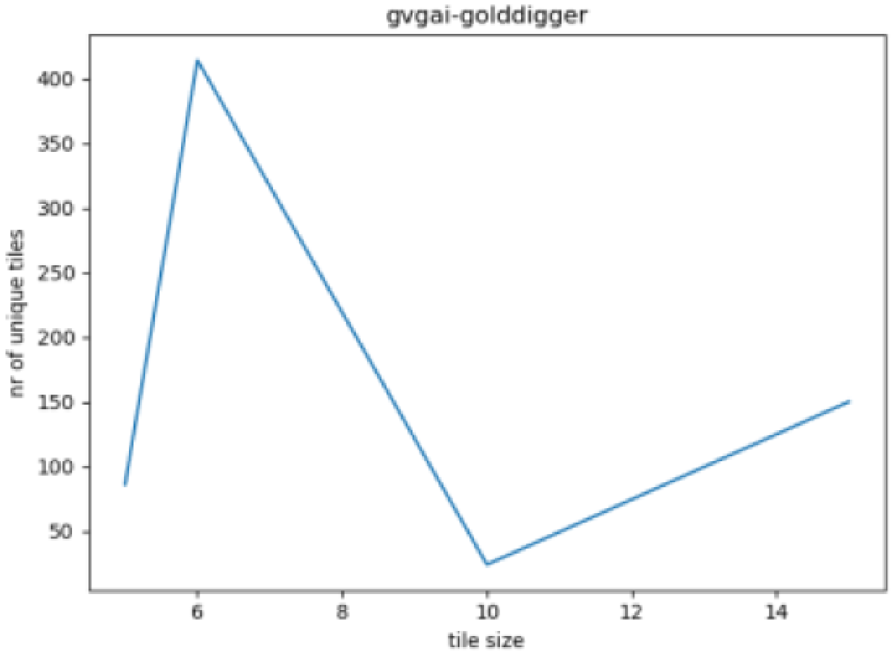
Which Tile-Size is Optimal?

Extracted tile-maps are compared given their number of unique tiles and their tile-size.

- a small number of unique tiles helps to keep the final model simple
- a large tile-size is desirable to reduce the size of the input matrix as much as possible

Algorithm:

- For each divisor of the original dimensions we extract a tile-map
- To assure interpretable models we chose the minimal amount of unique tiles
- This resulted in a tile-size of 10 for all games



Score Model

A score model is required to simulate the agent's reward.

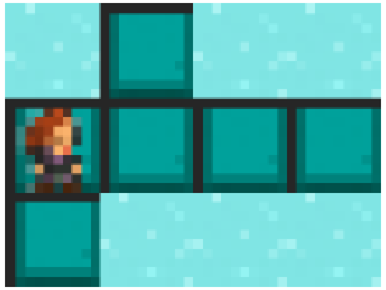
- Rewards in the GVGAI framework are bound to interactions between objects.
  - events are triggered when two bounding boxes overlap
  - which can result in the destruction/creation of objects and is associated with a reward

For each tile or object we extract the following values:

- its occurrence count in the state before the transition
- its occurrence count in the state after the transition
- the number of tiles/objects that have become this type
- the number of tiles/objects that are no longer of this type

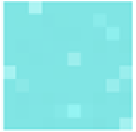
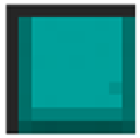
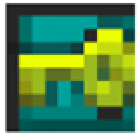
Score Model - Example

example state
waterpuzzle



predicted result
for action left

score model input

occurrence count	tile-type 			
before	6	4	1	1
after	6	5	1	0
created	0	1	1	0
destroyed	0	0	1	1

Active Learning Motivation

Training models using random exploration has shown to be inefficient.

- the agent often visits the same states and applies the same actions
- many patterns remain unexplored

We are aware of all possible patterns but gathering labels costs time and resources

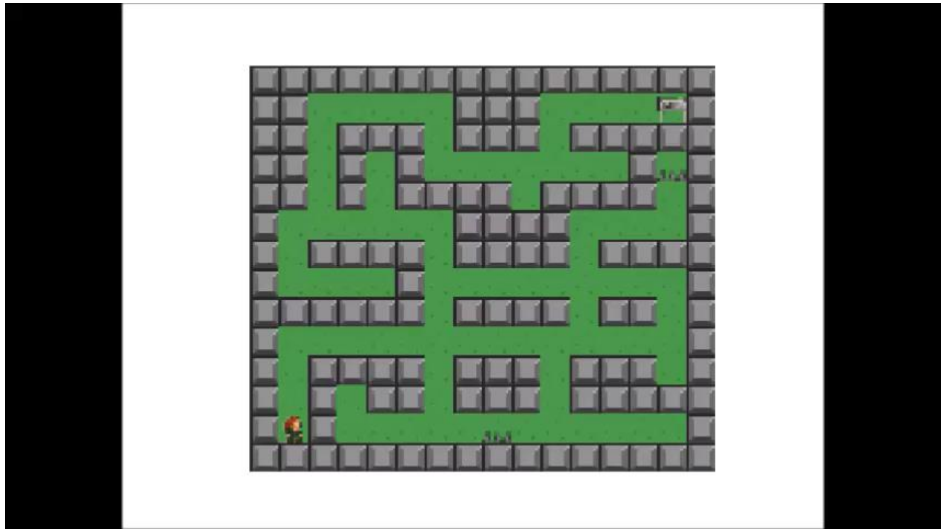
- **possible solution:** apply active learning techniques to increase the training efficiency

Active Learning Example

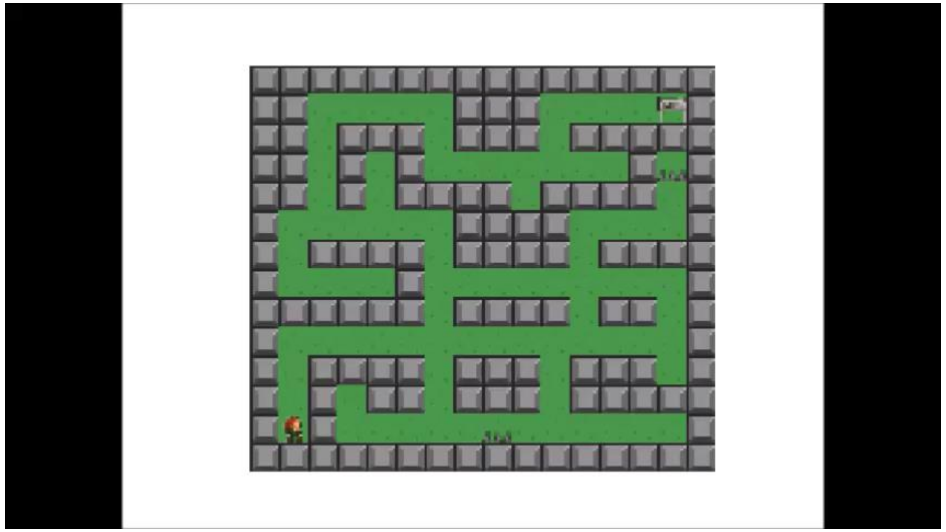
Implementation:

- During training the agent explores by choosing actions that yield the most unknown patterns
- In deterministic games, state-action pairs that have been explored will be simulated by the forward model to find interesting child states

Training (excluding symmetries)



Evaluation



Evaluation Setup

The evaluation is based on:

- three test games provided by the competition track, only the training levels are known
- six additional games have been chosen to evaluate the agent using unknown levels
- The agent has been trained using provided training levels and their symmetric counterparts

Learning-track games offer 2 levels to be trained and tested on:

- performance values of various agents were published for comparison
- the agents' training time is not limited by the competition rules

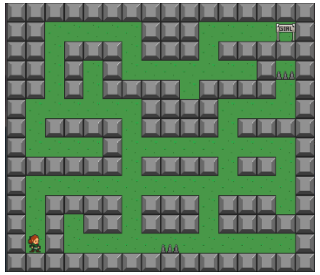
Our test games offer 2 levels to be trained on and 3 levels for evaluation:

- performance we compare the performance to search-based agents using the real forward model

Active Learning Results - Learning Track Games

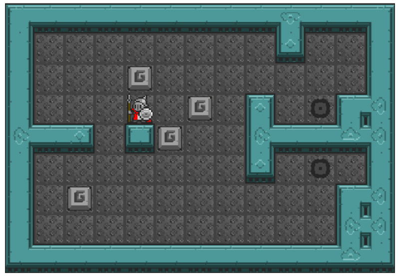
Game	Level	Proposed Agent	Reinforcement Learning			True Forward Model				
			DQN	A2C	PPO2	RS	RHEA	MCTS	OLETS	Random
waterpuzzle	0	**15.0**								3.5
waterpuzzle	1	**15.0**								2.5
treasurekeeper	0	7.75								0.75
treasurekeeper	1	**6.0**								0.75
golddigger	0	7.45								4.8
golddigger	1	4.4								8.2

Active Learning Results - Deterministic Games



Game	Level	Proposed Agent	True Forward Model RS	RHEA	MCTS	OLETS	Random
labyrinth	0	**1.0 / 1.0**	0.05 / -0.8	0.0 / -0.9	0.0 / -0.95	0.05 / -0.85	0.0 / -0.85
labyrinth	1	**1.0 / 1.0**	0.0 / -0.45	0.0 / -0.55	0.0 / -0.55	0.0 / -0.8	0.0 / -0.65
labyrinth	2	**1.0 / 1.0**	0.0 / -0.85	0.0 / -0.85	0.0 / -0.9	0.0 / -0.85	0.0 / -0.9
labyrinth	3	**1.0 / 1.0**	0.2 / -0.55	0.1 / -0.7	0.1 / -0.7	0.15 / -0.65	0.1 / -0.6
labyrinth	4	**1.0 / 1.0**	0.0 / -1.0	0.0 / -1.0	0.0 / -1.0	0.0 / -1.0	0.0 / -1.0

Active Learning Results – Non-Deterministic Games



Game	Level	Proposed Agent	True Forward Model				
			RS	RHEA	MCTS	OLETS	Random
sokoban	0	0.0 / 0.05	**0.0 / 0.15**	0.0 / 0.05	0.0 / 0.0	0.0 / 0.1	0.0 / 0.0
sokoban	1	0.0 / 0.2	**0.0 / 0.45**	0.0 / 0.15	0.0 / 0.2	0.0 / 0.3	0.0 / 0.15
sokoban	2	0.0 / 0.85	0.0 / 1.05	0.0 / 1.1	0.0 / 1.0	0.0 / 0.9	**0.0 / 1.2**
sokoban	3	0.0 / 0.25	0.0 / 0.5	0.0 / 0.5	**0.0 / 0.6**	0.0 / 0.4	0.0 / 0.2
sokoban	4	0.05 / 1.0	0.05 / 1.0	0.05 / 0.9	**0.2 / 1.15**	**0.2 / 1.15**	0.05 / 0.95

Conclusion

Deterministic Games:

- The agent was able to quickly learn a reliable model and play proficiently

Non-deterministic Games:

- Search-based methods struggled with the size of the possible state-space
- Probabilistic predictions have a low accuracy since the independency assumption is not fulfilled

Future Work:

- Explore the performance of other search-based methods with the trained models
- Map the model-building assumption (locality) to other models (e.g. DNN)

Thank you for your attention!

Interested in trying it yourself? Download the Code to this paper on Github
https://github.com/ADockhorn/Local-Forward-Model-Learning-for-GVGAI-Games

by Alexander Dockhorn and Simon Lucas
Email: {a.dockhorn, simon.lucas}@qmul.ac.uk