Deep learning in planning

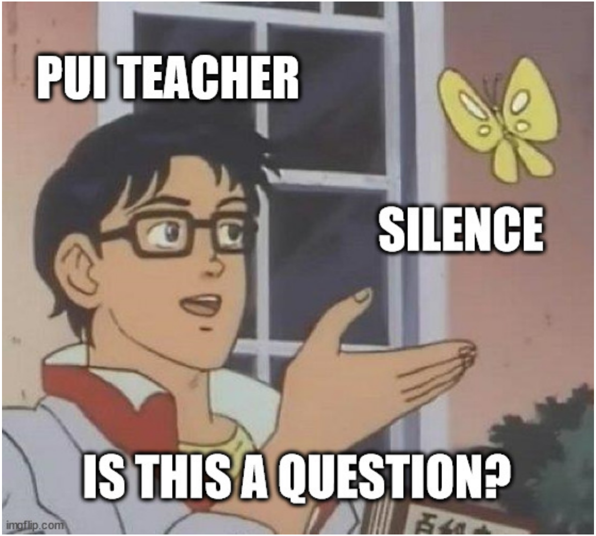
Assignment 1 consultations

Michaela Urbanovská

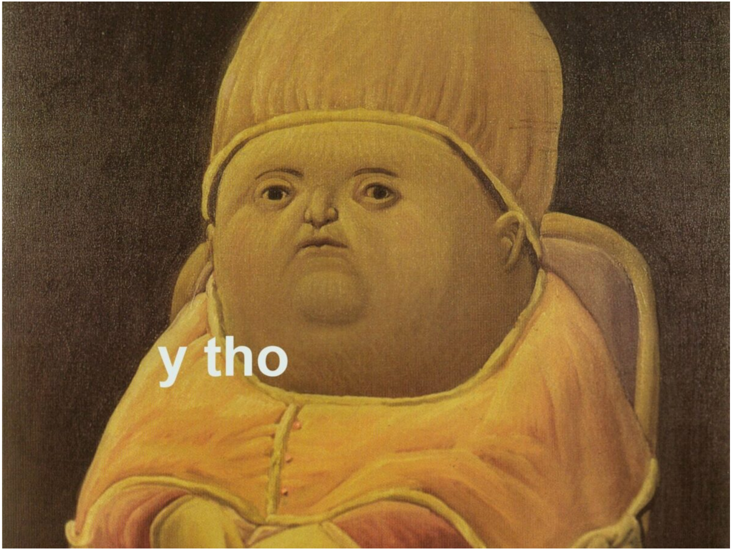
PUI Tutorial
Week 8

Lecture check

- Any questions regarding the lecture?



Deep learning in classical planning



Deep learning in classical planning

General intelligence

- To reach general intelligence we cannot just use machine learning
- H. Geffner's talk about Model-free learners and model-based solvers [5]
- Similar Kahneman's mind model with System 1 and System 2 in [7]
- *What I do*
  - **model-based solver** classical planning
  - **model-free learner** deep learning

Introduction to Planning

- Problems typically modeled by hand
- Standard languages / representation (PDDL, PPDDL, STRIPS, FDR, ...)
- Solved by off-shelf planners

Introduction to Planning

Planning problem in PDDL

- Domain definition
  - Predicates
  - Actions - parameters, preconditions, effects
- Problem definition
  - Objects
  - Initial state - set of propositions
  - Goal state specification - set of propositions

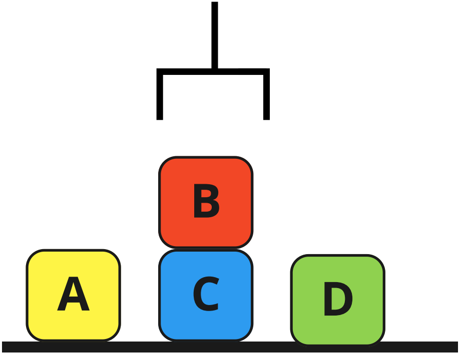
Introduction to Planning

```
(define (domain blocksworld)
  (:requirements :strips)
  (:predicates (on ?x ?y)
           (ontable ?x)
           (clear ?x)
           (handempty)
           (holding ?x)
           )

  (:action pick-up
         :parameters (?x)
         :precondition (and (clear ?x) (ontable ?x) (handempty))
         :effect
         (and (not (ontable ?x))
           (not (clear ?x))
           (not (handempty))
           (holding ?x)))

  (:action put-down
         :parameters (?x)
         :precondition (holding ?x)
         :effect
         (and (not (holding ?x))
           (clear ?x)
           (handempty)
           (ontable ?x)))
  (:action stack
         :parameters (?x ?y)
         :precondition (and (holding ?x) (clear ?y))
         :effect
         (and (not (holding ?x))
           (not (clear ?y))
           (clear ?x)
           (handempty)
           (on ?x ?y)))
  (:action unstack
         :parameters (?x ?y)
         :precondition (and (on ?x ?y) (clear ?x) (handempty))
         :effect
         (and (holding ?x)
           (clear ?y)
           (not (clear ?x))
           (not (handempty))
           (not (on ?x ?y)))))
```

```
(define (problem p01-blocksworld)
(:domain blocksworld)
(:objects A B C D)
(:INIT
    (ontable A) (ontable C) (ontable D) (on B C)
    (clear A) (clear B) (clear D) (handempty)
)
(:goal (AND
    (on C D) (on B C) (on A B)
    )
)
)
```



Planning problem represented by STRIPS

$\Pi = \langle F, O, s_i, s_g, c \rangle$

- F - set of facts that can hold in the world
- O - set of operators which can be used to transform the world
- s_i - fully defined initial state of the world
- s_g - goal condition that holds in every goal state
- c - cost function which gives cost to every operator

State

Every state $s \in S$ is a set of facts from F.

Operator

Every operator is a tuple that contains preconditions, add effects and delete effect for the given operator

$$o = \langle pre(o), add(o), del(o) \rangle$$

Operator o is applicable in state s if $pre(o) \subset s$. By applying o in s we get state s'

$$s' = (s \backslash del(o) \cup add(o))$$

PDDL and STRIPS action

```
PDDL
(:action pick-up
      :parameters (?x)
      :precondition (and (clear ?x) (ontable ?x) (handempty))
      :effect
      (and (not (ontable ?x))
        (not (clear ?x))
        (not (handempty))
        (holding ?x)))

STRIPS
pickup(A) = <{clear(A), ontable(A), handempty},
            {holding(A)},
            {clear(A), ontable(A), handempty}>
```

Transition system

$\Sigma = \langle S, A, \gamma, c \rangle$

- S - set of states
- A - set of actions
- γ - state transition function
- c - cost function

Solving a planning problem means looking for a path in the graph induced by the transition system.

- Forward search
- Backward search
- Bidirectional search

- Most likely any search can use a heuristic function
- Heuristic function makes the search **informed**

Heuristic function

Heuristic function $h(s)$ maps any state s to a value that represents path length from s to a goal state.
Function that maps each state s to the length of shortest path from s to a goal is $h*$ which is the perfect or optimal heuristic.

Deep Learning in Classical Planning

- Many different possible applications
  - grounding
  - heuristic computation
  - model learning
  - planning in latent space
  - many more...
- Data which is not noisy
- Relatively small data sets
- Hard to compare with existing approaches

Deep Learning in Classical Planning

Examples of research

- Framework inspired by Kahneman's work [3]
- Learning policies and heuristics from images [6]
- Planning with images in latent space [1], [2]
- Using neural networks to learn heuristic functions [4]
- ...or literally anything that I did

Deep Learning in Classical Planning

Drawbacks of many of these approaches

- Input size or format
- Domain-independence / generalization abilities
- Size of the network
- Speed of the evaluation
- Time required for training
- Overall results

Ongoing research

Ongoing research

Different domains

- Maze traversal problem *(vanilla, multi-goal, multi-agent)*
- Sokoban
- ...omw to domain-independence as well

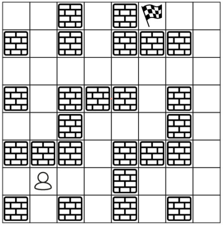
Different problems

- Learning transition function
- Learning heuristic
- Planning for grid domains
- Creating machine learning friendly problem representation
- Creating domain-independent architectures

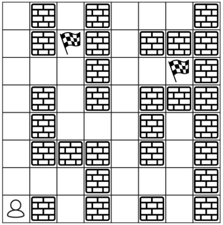
Learning transition system + heuristic function

- State space + state-transition function
- **Expansion network** that works with graphic representation
- Heuristic function
- **Heuristic network** that works with graphic representation

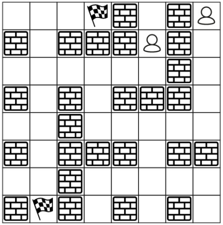
Used domains



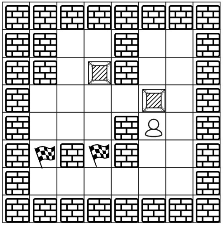
Single-agent maze



Multi-goal maze



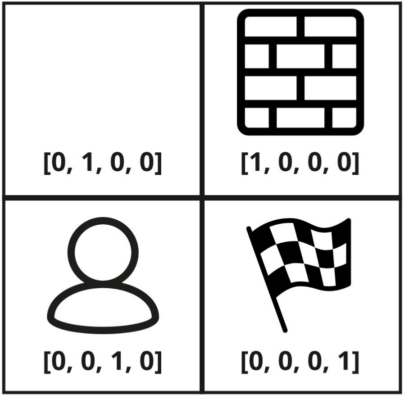
Multi-agent maze



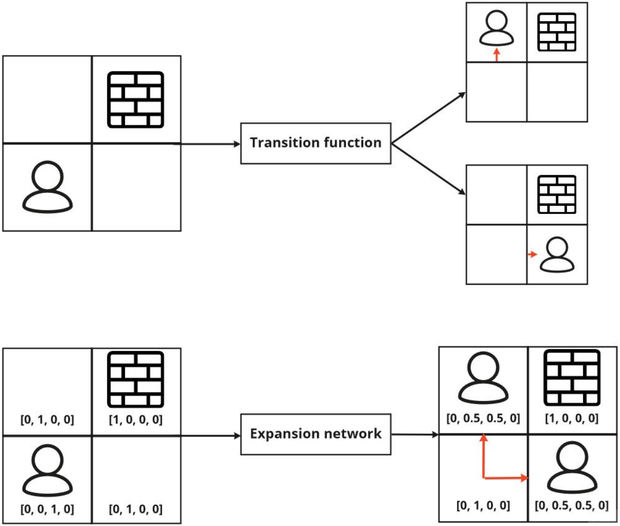
Sokoban

Learning transition system + heuristic function

- Grid domains (so far...)
- One-hot encoding of the entities on cells
- Convolutional + recurrent neural networks
- **Scale-free** architectures

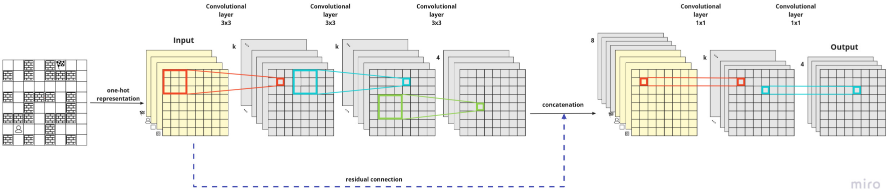


Expansion network



Expansion network

- Convolutional neural network
- 4-neighborhood movement possible
- 3×3 convolutional window to see the surroundings of the agent
- residual connection in the architecture to not loose initial information

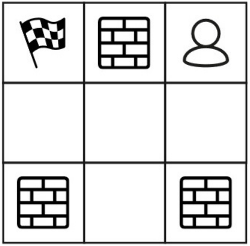
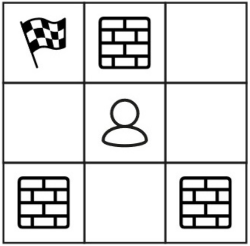
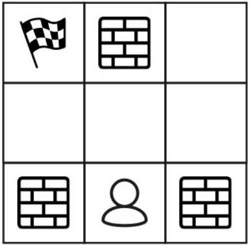


Heuristic network

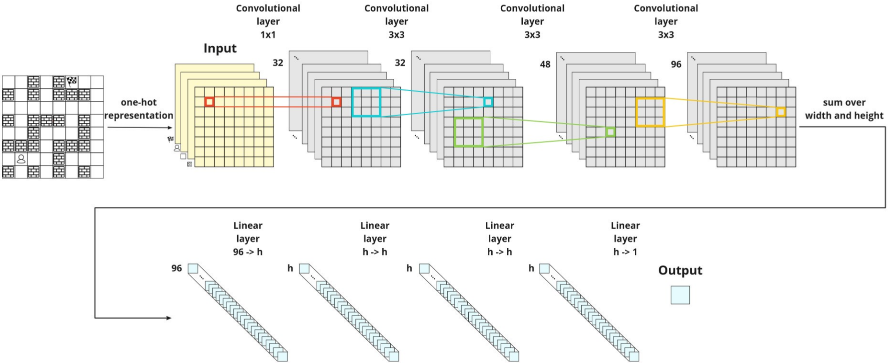
- 3 architectures
- **CNN** - convolutional neural network (most simple)
- **CNN_att** - convolutional neural network using attention
- **RNN** - reasoning recurrent network using MAC cell
- Inspiration in landmarks, relaxations, abstractions...
- Each architecture has intuition

Heuristic network - loss function

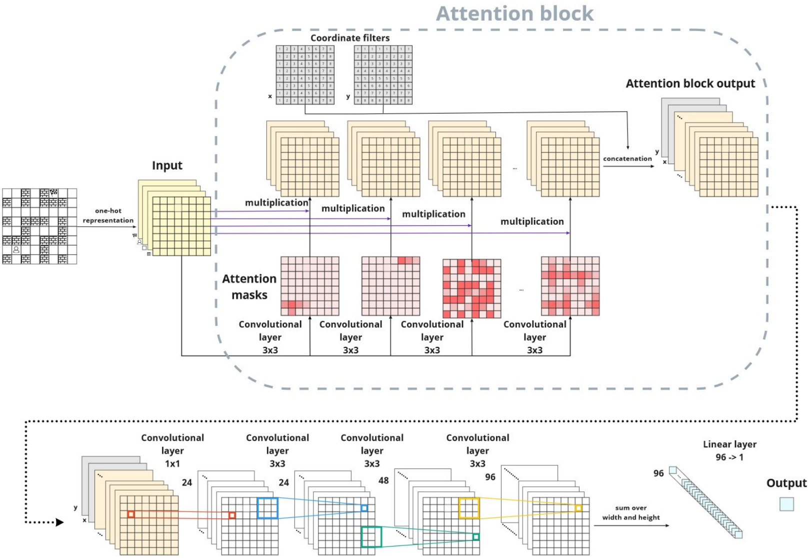
- We're learning **monotonocity**
- Property of a good heuristic
- sample + label pairs aren't enough anymore
- one instance with multiple agent placements

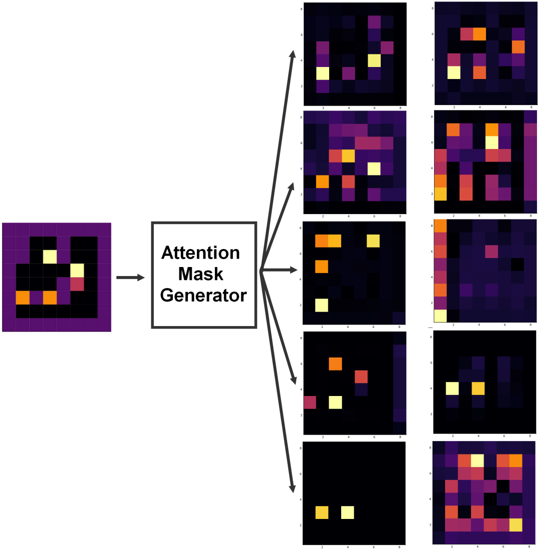
Heuristic network - CNN



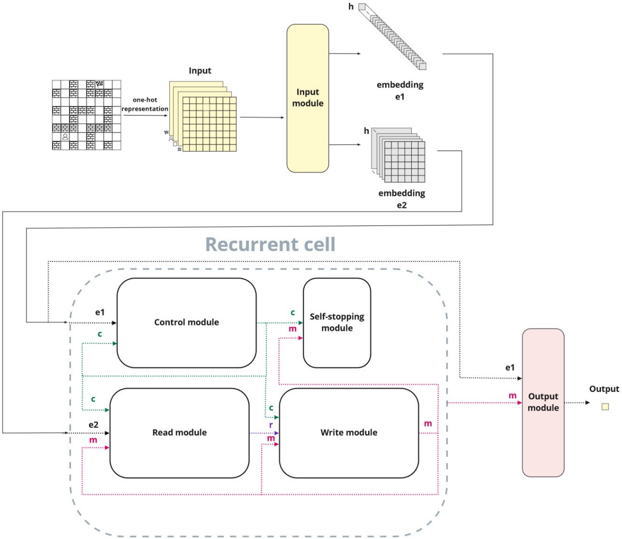
Heuristic network - CNN_att



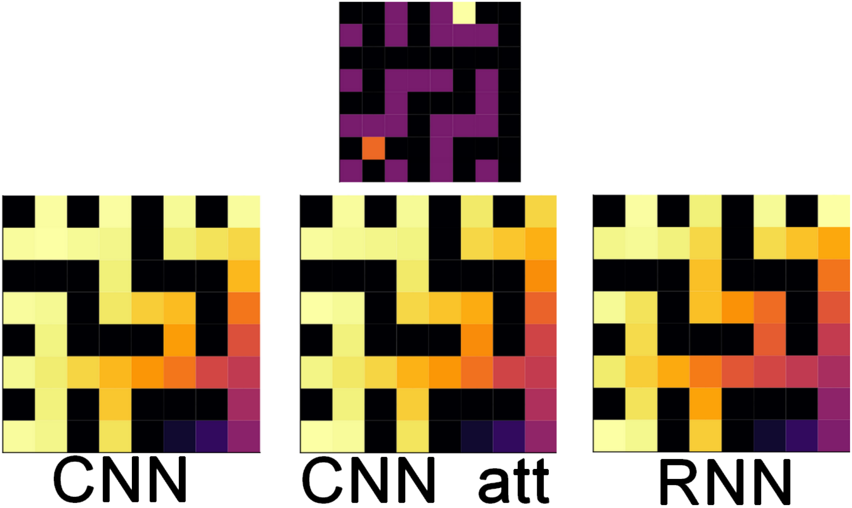
Heuristic network - CNN_att Sokoban attention masks



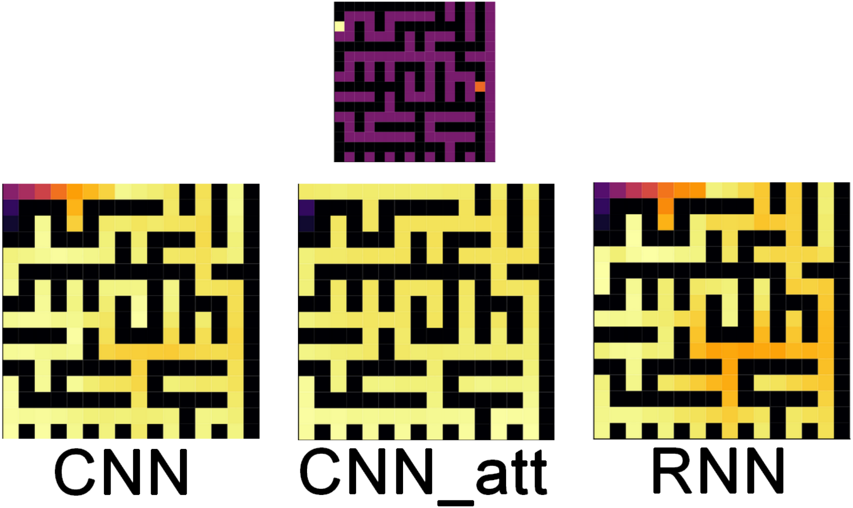
Heuristic network - RNN



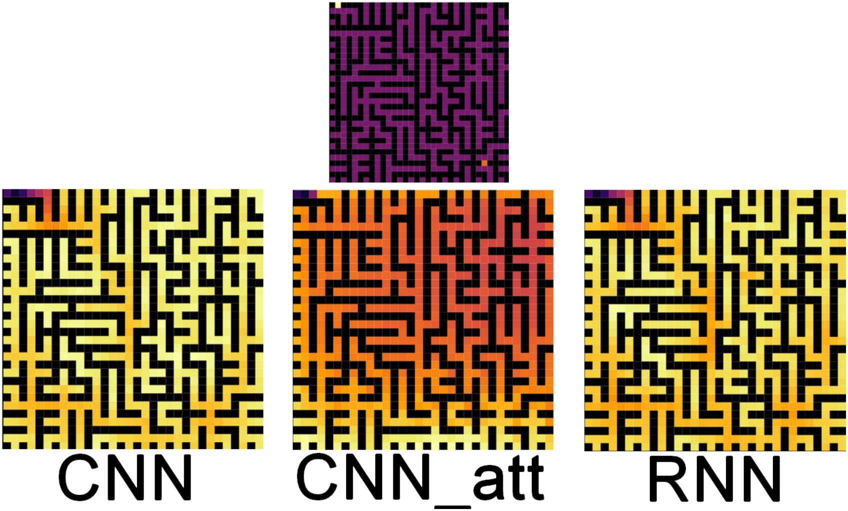
Architecture comparison - 8×8



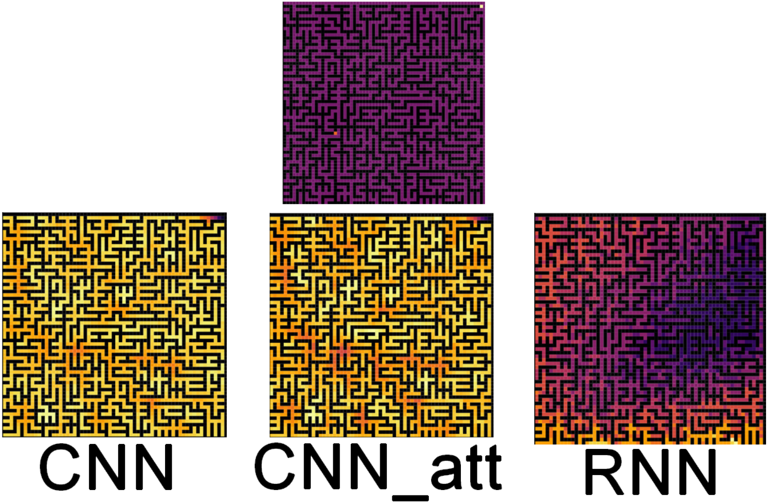
Architecture comparison - 16 × 16



Architecture comparison - 32×32



Architecture comparison - 64×64



Experiments

- We tested all architectures against Euclidean distance, h^{LM-Cut} and h^{FF} heuristics
- Results on par on small domains
- Time advantage in large complex domains
- Less informed values in large state-spaces
- Slower expansion can slow down the search too much

Learning heuristic function using CSRN

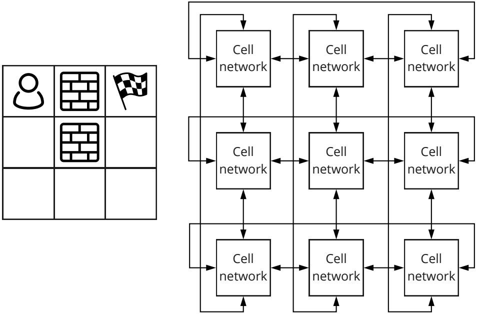
Cellular Simultaneous Recurrent Neural Network (CSRN)

- Recurrent architecture
- Weight sharing
- Input represented by grid
- Scalable solution for input of any size
- Originally used to solve maze navigation problem

Learning heuristic function using CSRN

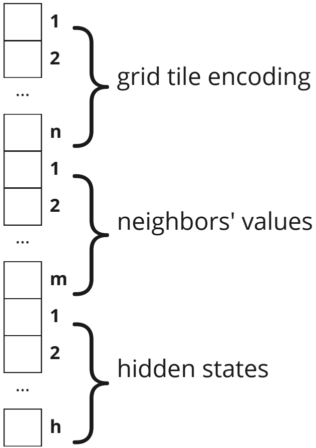
CSRN architecture consists of **cell networks** which share weights.

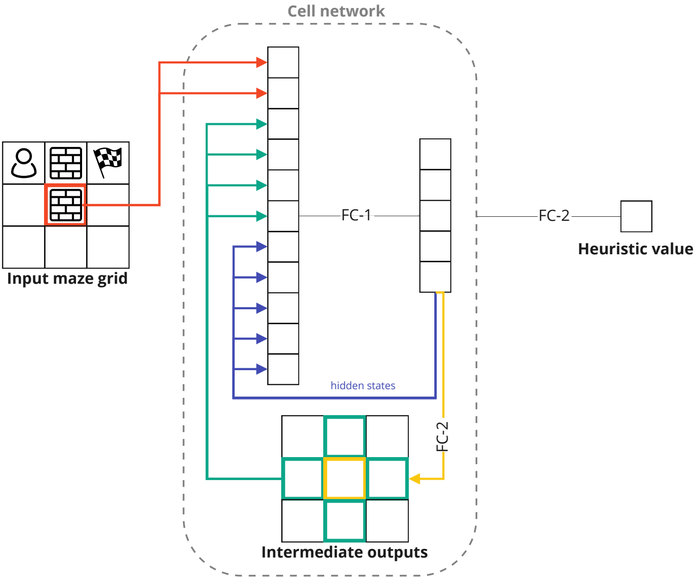
- **Cell network**
  - Small recurrent network
  - Operating over **one** grid cell
  - Same set of weights for every **cell network**
  - Sending intermediate results with neighboring cell networks



Input

- One grid cell represented by one vector
  - n - length of grid tile encoding
  - m - number of neighbors
  - h - number of hidden states




Cell network
Input maze grid
FC-1
FC-2
Heuristic value
hidden states
FC-2
Intermediate outputs

CSRN output

- CSRN generates heuristic value for each grid cell
- Interpretation with respect to agent's coordinates for both domains

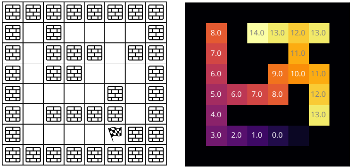


Figure: CSRN Output for Maze Domain

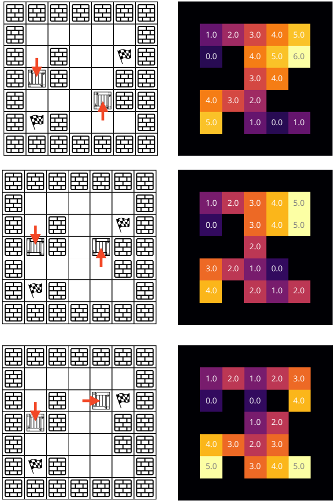
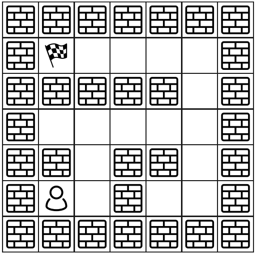


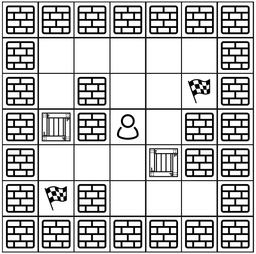
Figure: CSRN Output for Sokoban

Training and experimental setup

- *Problem domains:* maze, Sokoban
- *Training data:* small number of small exhaustively solved problem instances
  - *Maze:* 5 instances of size 5×5
  - *Sokoban:* 1 instance of size 3×3 with one box
- *Optimization method:* Bayesian optimization
- *Objective function:* number of incorrect decisions in the search algorithm



Maze



Sokoban

Experiment results

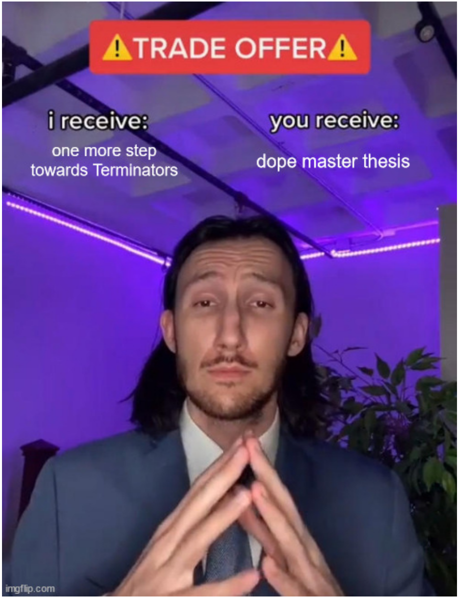
- CSRN is capable of scaling to larger problem instances
- CSRN can be used on different grid domains
- Sokoban results exceeded expectations
- Results showed ability to generalize and perform well on larger / more complex problem instances

Story goes on...

Since then we are working on

- Different CSRN settings ("3D", variable number of recurrent iterations, differentiability, loss functions)
- Domain-independent CSRN-like architecture for STRIPS problems
- Learning heuristic analogical to potential heuristic
- Looking for more grid domains that can be used with this architecture
- ... many more things

Story goes on...



The End

Feedback form link

Citations I

Masataro Asai and Alex Fukunaga.
Classical planning in deep latent space: From unlabeled images to PDDL (and back).
In Tarek R. Besold, Artur S. d'Avila Garcez, and Isaac Noble, editors, *Proceedings of the Twelfth International Workshop on Neural-Symbolic Learning and Reasoning, NeSy 2017, London, UK, July 17-18, 2017*, volume 2003 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2017.

Masataro Asai and Alex Fukunaga.
Classical planning in deep latent space: Bridging the subsymbolic-symbolic boundary.
In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Di Chen, Yiwei Bai, Wenting Zhao, Sebastian Ament, John M. Gregoire, and Carla P. Gomes.
Deep reasoning networks: Thinking fast and slow.
CoRR, abs/1906.00855, 2019.

Hung-Che Chen and Jyh-Da Wei.
Using neural networks for evaluation in heuristic search algorithm.
In Wolfram Burgard and Dan Roth, editors, *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011*. AAAI Press, 2011.

Hector Geffner.
Model-free, model-based, and general intelligence.
In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 10–17. ijcai.org, 2018.

Edward Groshev, Aviv Tamar, Maxwell Goldstein, Siddharth Srivastava, and Pieter Abbeel.
Learning generalized reactive policies using deep neural networks.
In *2018 AAAI Spring Symposium Series*, 2018.

Citations II

Daniel Kahneman.
Thinking, fast and slow.
Macmillan, 2011.