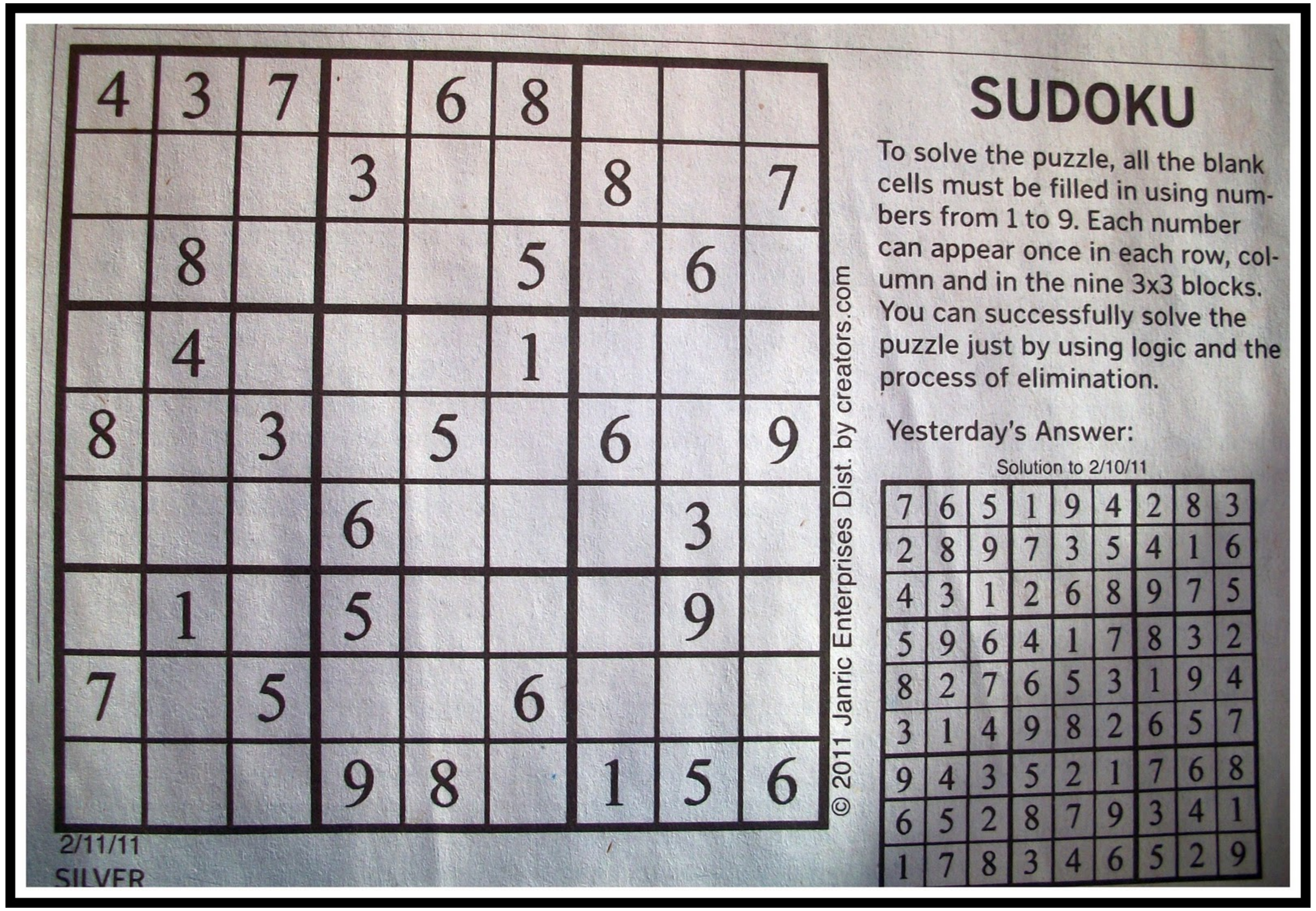
Learning CSPs via ILP solvers

Rishabh Ranjan, Yatin Nandwani, Mausam, Parag Singla

BTP Mid-Term Project Presentation

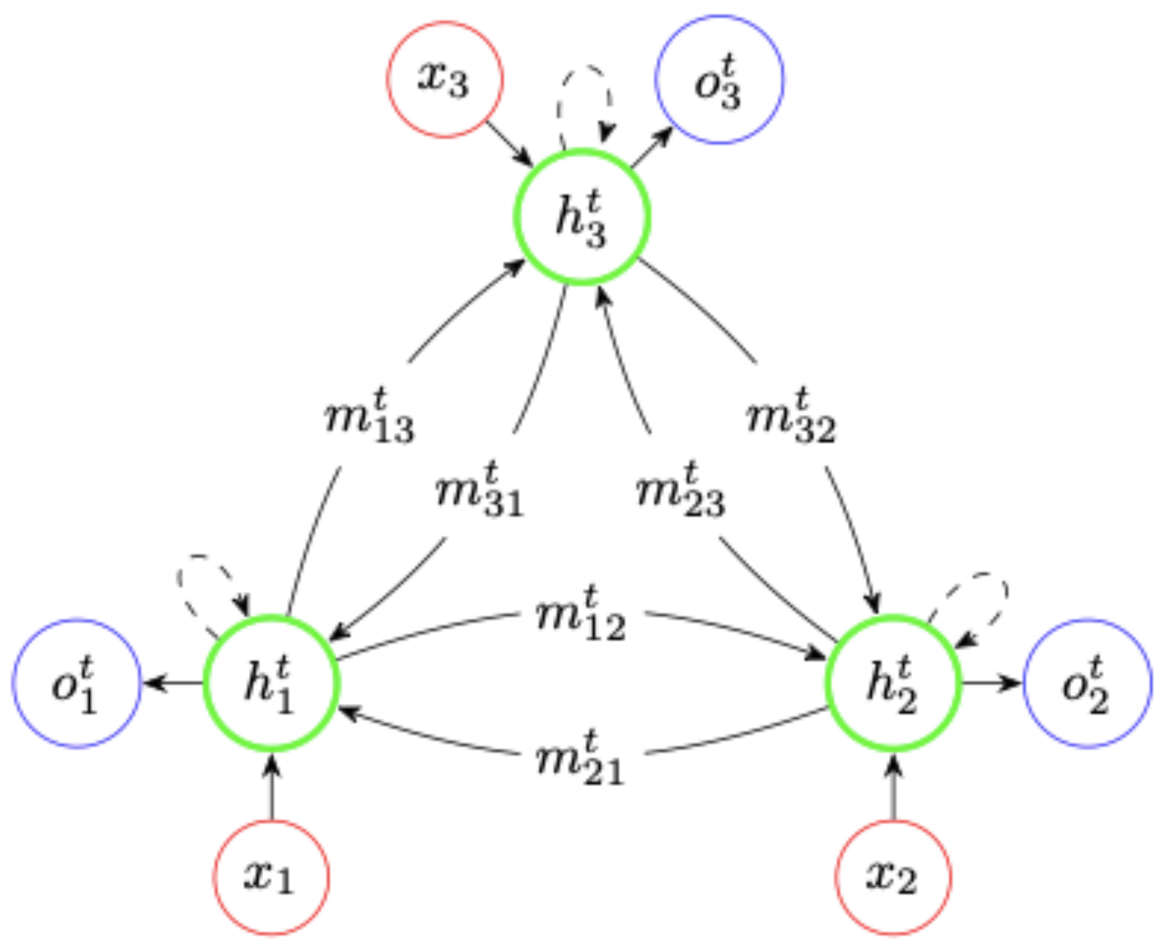
Motivation

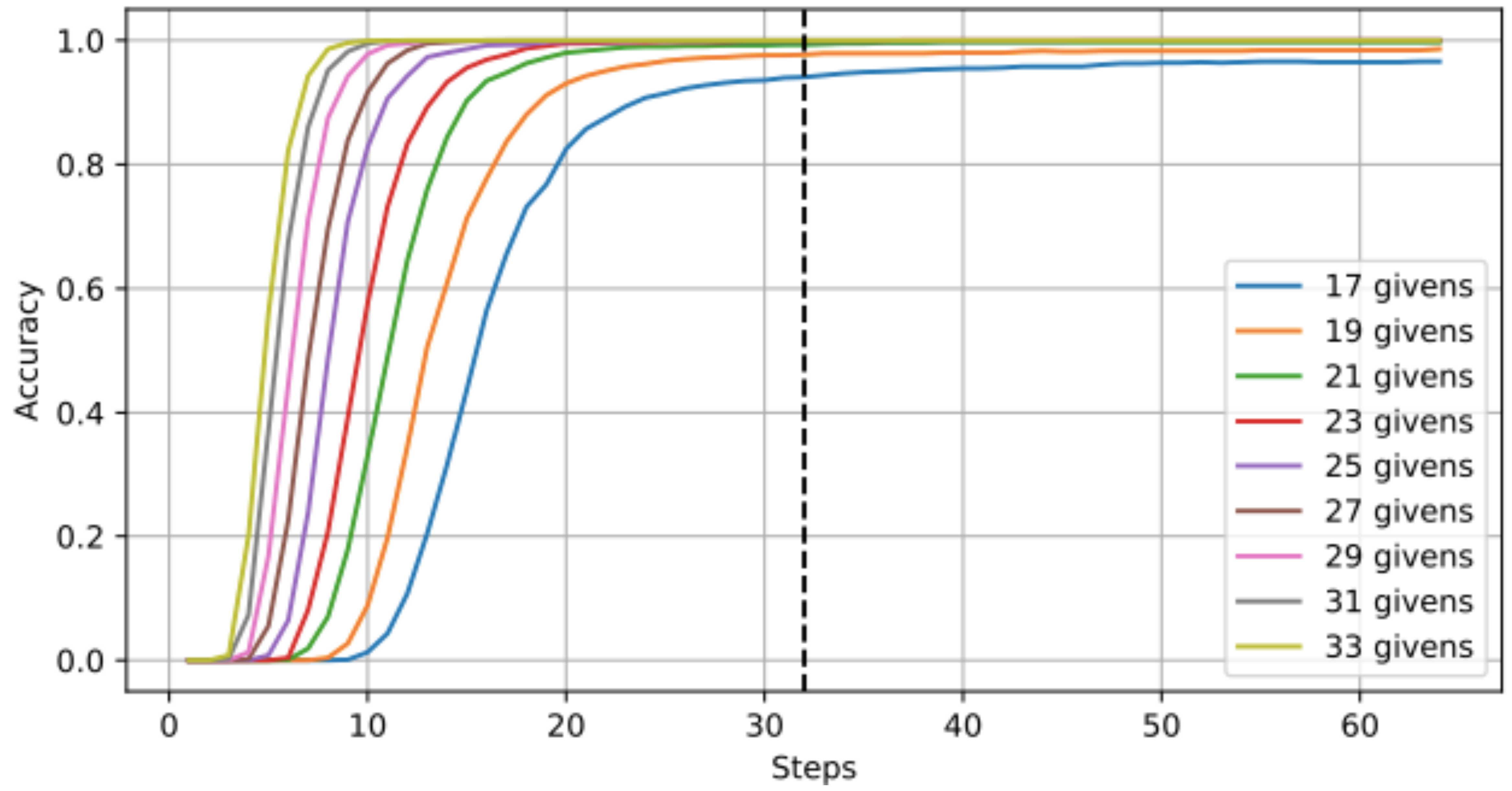
- The learning aspect
- CSPs in the real world:
  - industrial scheduling
  - combinatorial optimization
  - program verification
  - … and lots more
- Can neural networks learn CSPs from examples?



Approach 1: Neural Reasoner

Recurrent Relational Networks, Palm et. al., NeurIPS 2018

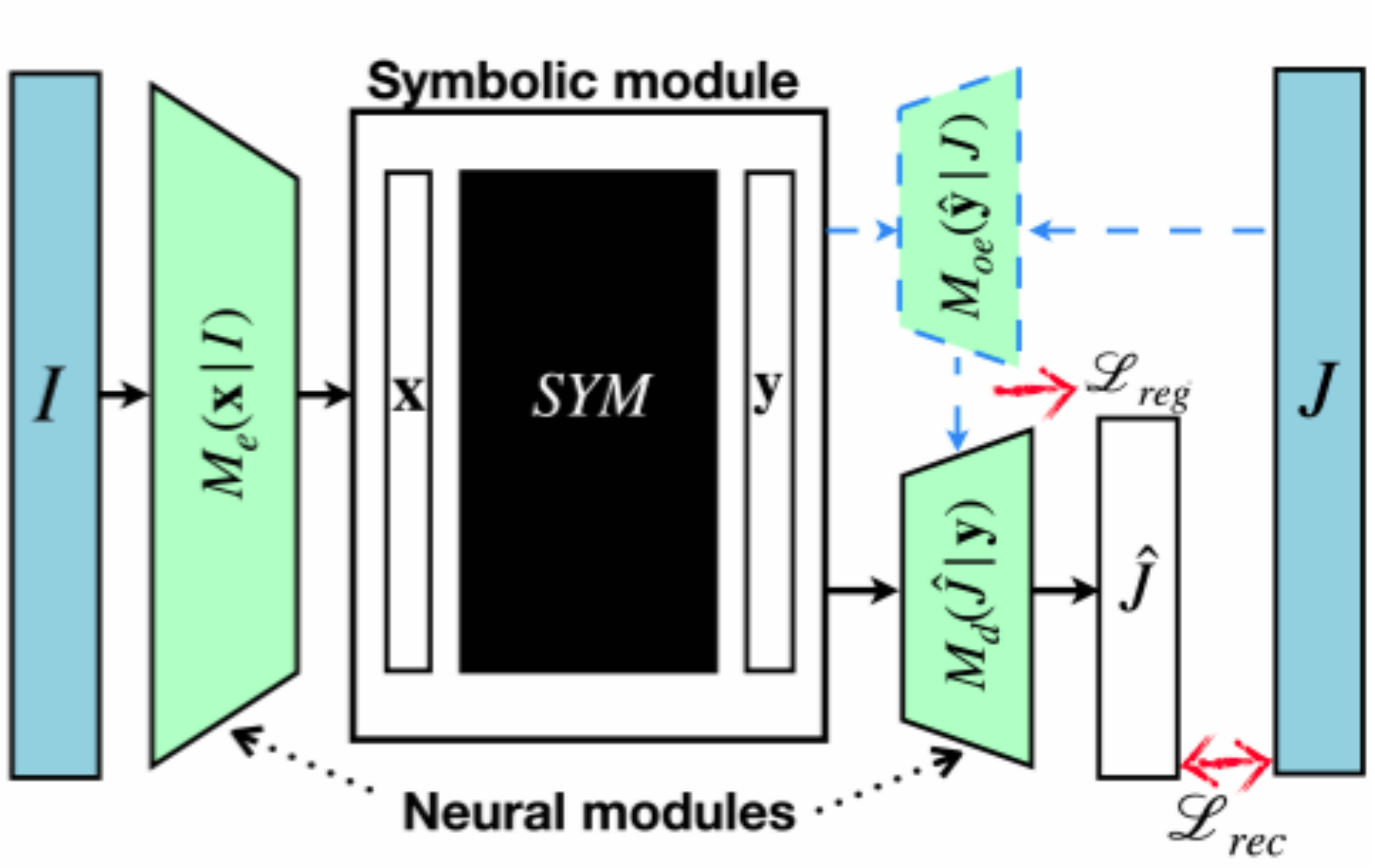




- Learns to reason from scratch
- Problem learning and solution learning are coupled together
- No match for decades old symbolic approaches

Approach 2: Symbolic Reasoner + RL

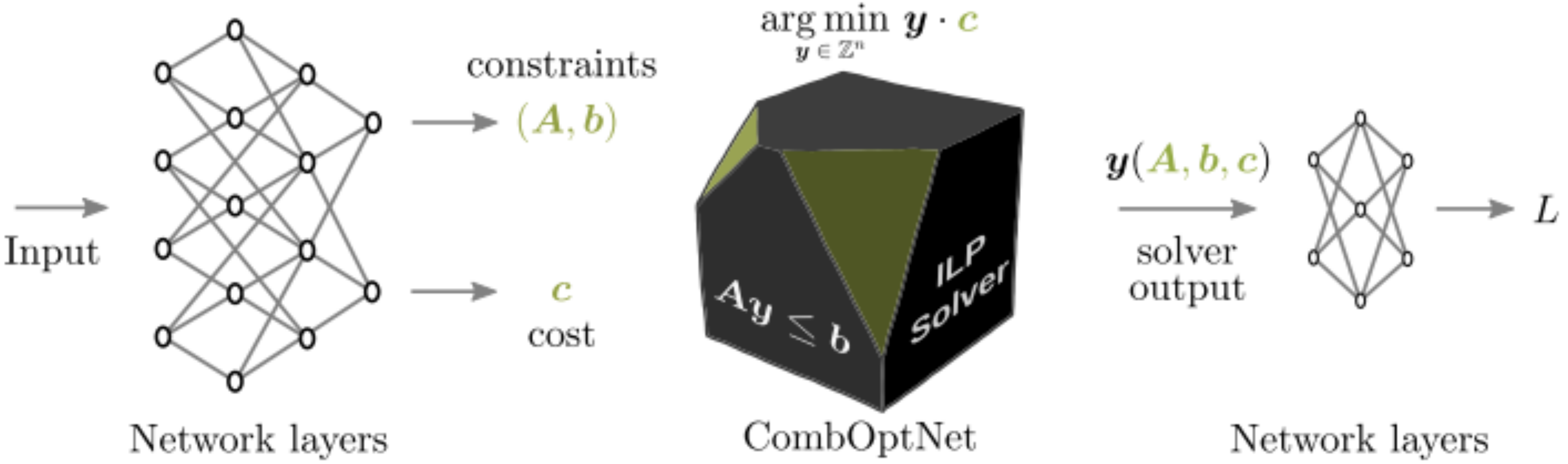
End-to-End Neuro-Symbolic Architecture for Image-to-Image Reasoning Tasks, Agarwal et. al., unpublished, 2021



- Difficult to train: sparse rewards
- Problem specific solver: intractable action space for general-purpose solver

Approach 3: Symbolic Reasoner + Backprop

CombOptNet: Fit the Right NP-Hard Problem by Learning Integer Programming Constraints, Paulus et. al., ICML 2021



- Backprop via informative gradients for black-box ILP solvers
- Any problem in NP is reducible to ILP (NP-completeness)
- Industry-grade solvers available: Gurobi

Limitations of Demonstrated Tasks

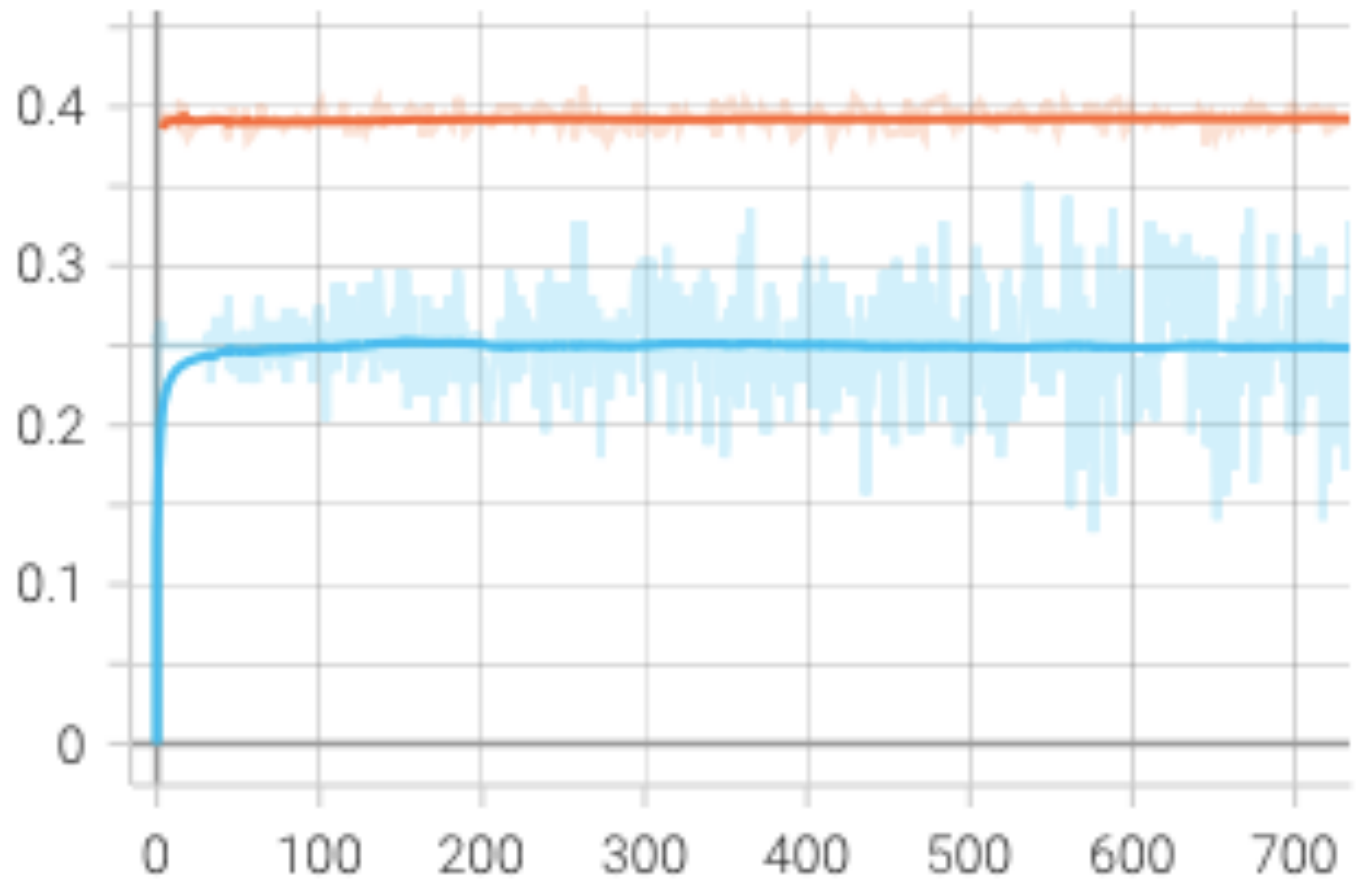
- Naive parametrisation of constraint matrix
- **Synthetic:** random dense constraints
- **Small:** max 8 constraints over 16 variables
- **Input dependence:**
  - only 1 setting, with only 1 constraint
  - static constraints in most tasks
- **Instance-size dependence**
- **Slow training:** 6 hrs for 4000 weight updates on one task (prior art took 1 hr)

Real-World CSPs

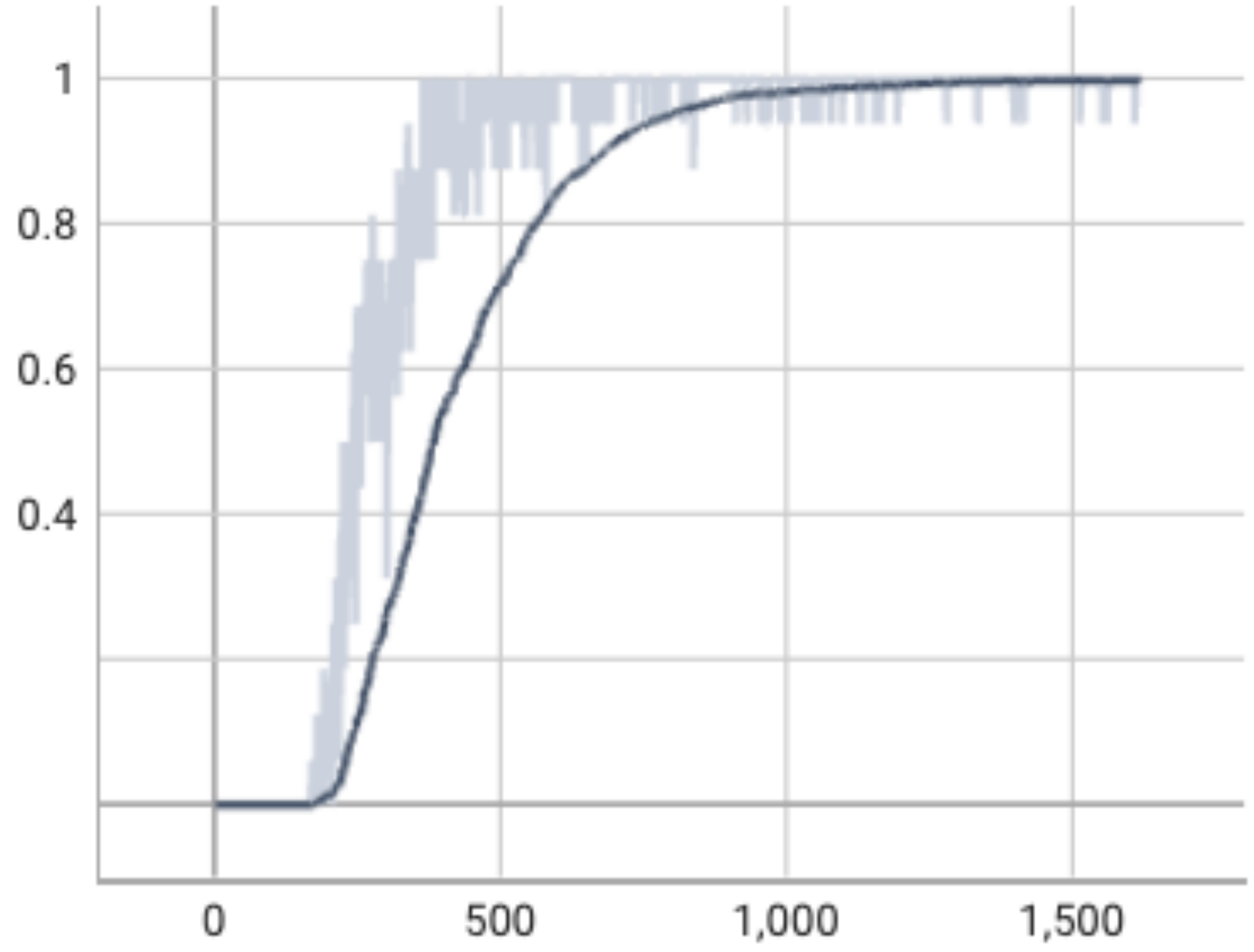
(Toy CSP but representative)

- ILP for 4x4 Sudoku: 82 constraints over 64 variables
- Input dependent constraints
- **Empirical Results:**
  - 4x4, static constraints only:
    - 100% accuracy
  - 4x4, fully learnable / 9x9, static constraints only:
    - trivial baseline accuracy

Accuracy Curves



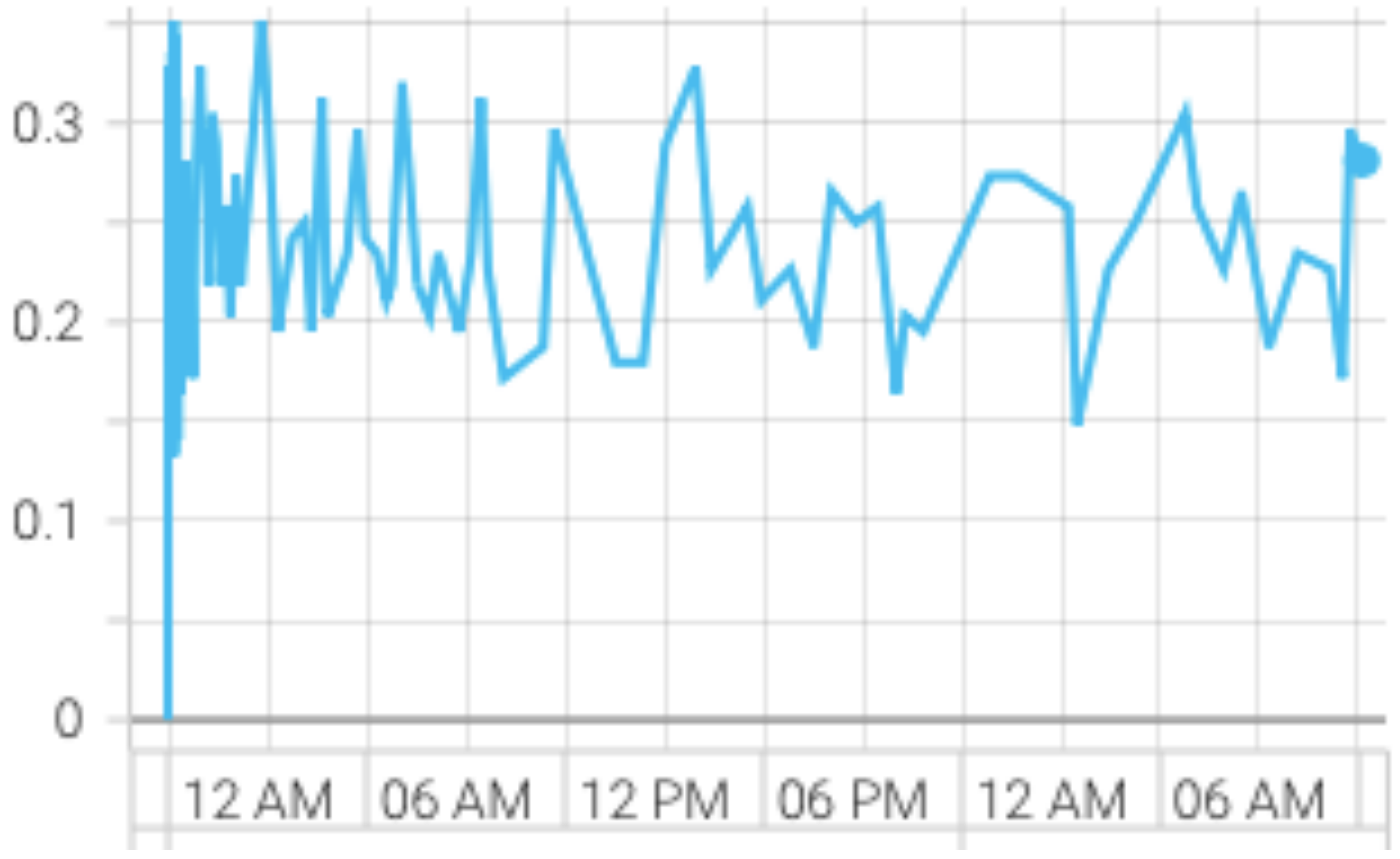
Smoothed Digit Accuracy vs Training Steps
(BLUE: 4x4 sudoku, fully learnable,
ORANGE: 9x9 sudoku, only static constraints learnable):
stuck as trivial baseline accuracies of 25% and 40%



Smoothed Board Accuracy vs Training Steps
(4x4 sudoku, only static constraints learnable):
100% generalisation accuracy

Technical Challenge 1: Solver Interaction

- Forward pass calls ILP solver
- Randomly learnt constraints at training
- Unaligned to solver heuristics
- Explosive solving times
- Extremely slow training
- GPU acceleration is useless



Digit Accuracy vs Training Steps
(4x4 sudoku, fully learnable):
Observe how training steps slow down.

Tricks to Speed Up Training

- **Initialisation** with trivially feasible constraints
- Adaptive **solver timeouts**: low timeout initially, increase gradually
- **Solution hinting:**
  - solution of same training instance from last epoch
  - gold solution at later stages of training
  - advanced **mini-batching**: consecutive batches can have same instances
- Priors can help generate easier constraints

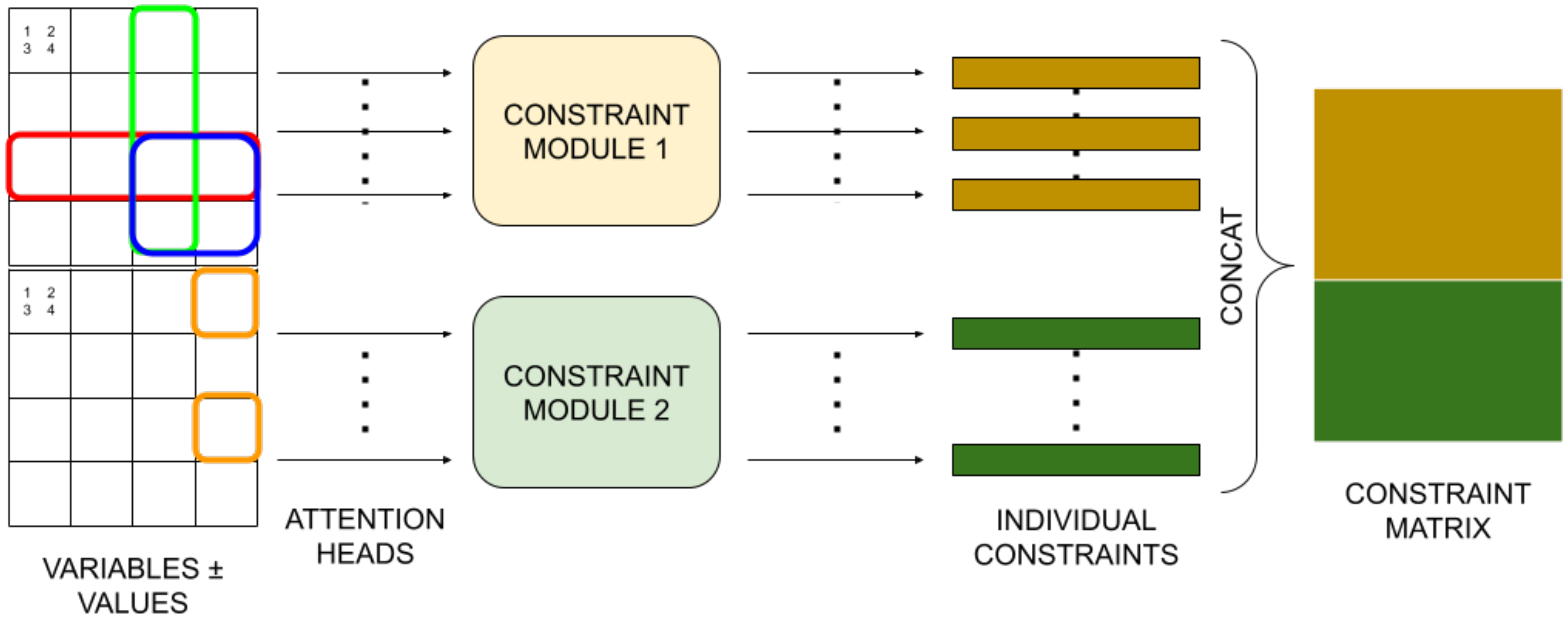
Technical Challenge 2: Parameterisation

- Each coefficient is learnt independently
- Example 1:
  - `ONE-HOT` constraint: $x + y + z = 1$ (x, y, z in $\{0, 1\}$)
  - ILP form: (1) $x + y + z \leq 1$, (2) $(-x) + (-y) + (-z) \leq -1$
- Example 2:
  - `ALL-DIFF` constraint:
    - 9 rows, 9 columns, 9 boxes

The Real-World Redemption

- Priors:
  - **Sparsity:** ~99% for sudoku
  - **Lifting:** instantiations of the same constraints over variable permutations
  - **High-level constraints:**
    - Google OR Tools: `AddAllDifferent`, `AddReservoirConstraint`, `AddAllowedAssignments`, `AddBoolOr`, `AddMaxEquality`
    - Microsoft Z3 SMT Solver: `AtLeast(k), AtMost(k)`
  - **Compositionality:** high-level from low-level constraints
- Heavily engineered solvers: Heuristics and optimisations for real-world instances

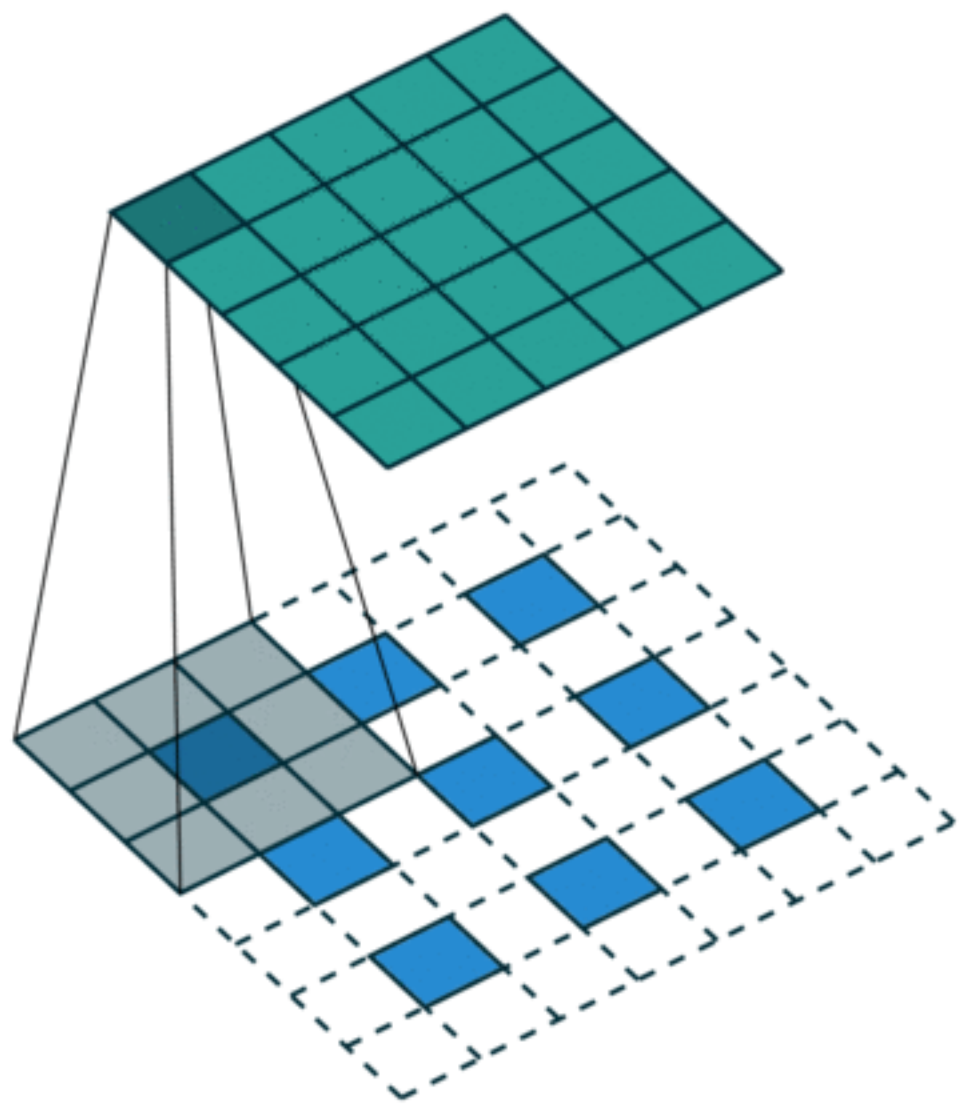
Proposed Architecture for Parameterisation



- **Attention** over variables (and values for input-dependence) as taking **soft subsets**
- **Learnable** or **hand-crafted** constraint modules
- **Permutation invariance:**
  - limitation in general
  - practically useful as high-level constraints tend to be permutation invariant

Alternative Solutions / Baselines / Directions

- **L1 regularisation:** sparsity prior
- **Deconvolution networks:** weight sharing in generation
- Restore generality using sequence selector modules
- Add a **compositionality** prior by going deeper



A deconvolution filter

Desired / Expected Benefits

- **Data efficiency:** learn only to formulate the problem
- **Faster forward pass:** priors aligned with ILP heuristics
- **Strong generalisation:** train on easy instances, test on hard instances
- **Domain knowledge:** enforce hard constraints directly
- **Instance size generalisation:** train on small instances, test on larger instances

The Plan Ahead

- DONE:
  - Problem formulation
  - Major implementation
  - Baseline Experiments
- TODO:
  - Thorough experiments on proposed architecture
  - Iterative development of more effective ideas

Thank You!

Questions?