A Solver-free Learning Framework for ILP

Rishabh Ranjan, Yatin Nandwani, Mausam, Parag Singla

(submitted to **NeurIPS 2022**)

BTP2 Final Presentation – 31 May 2022

Introduction

Motivation

<u>Task</u>

Perception → Reasoning

//

Pure Reasoning

<u>Model</u>

Neural Module → Symbolic Solver

//

Learnt Instance → Symbolic Solver

Integer Linear Programming

$x \rightarrow$ input

$A(x; \theta_A) \rightarrow$ constraint matrix

$b(x; \theta_b) \rightarrow$ bias vector

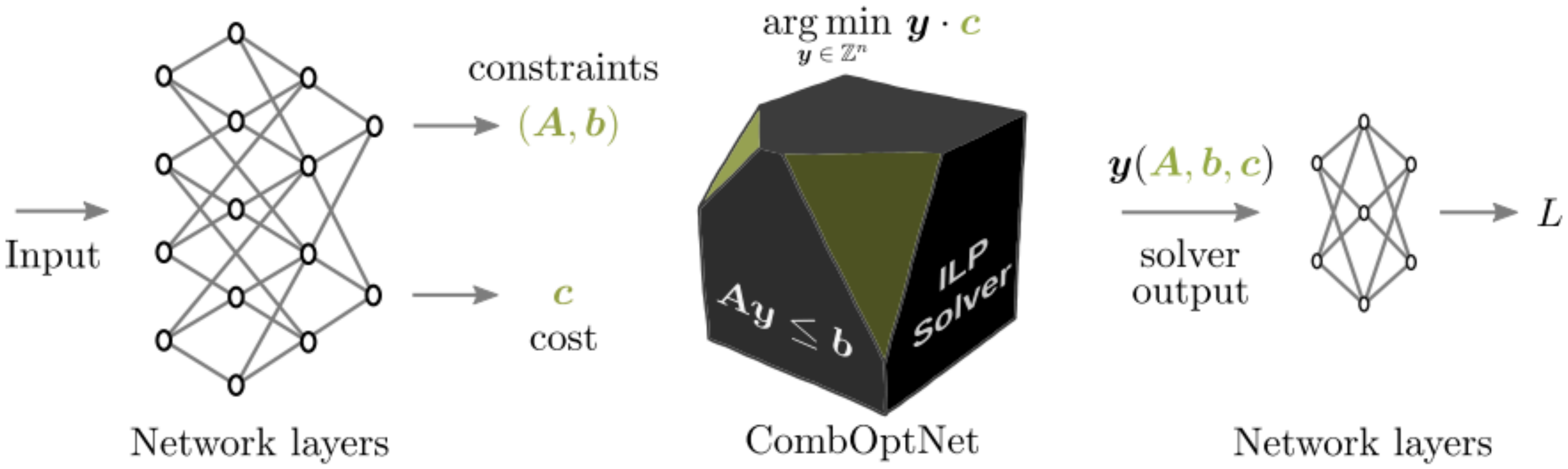
$c(x; \theta_c) \rightarrow$ cost vector

$y(A, b, c) \rightarrow$ solution vector / output

$n \rightarrow$ number of variables

$\min c \cdot y$ such that $Ay + b \geq 0, y \in \mathbb{Z}^n$

Related Work: CombOptNet[1]



Invokes solver in forward pass

Not scalable to large constraint matrices

[1] Paulus, Anselm et al. "CombOptNet: Fit the Right NP-Hard Problem by Learning Integer Programming Constraints." ICML (2021).

Experiments

Visual Sudoku: Setup

Input: grid of digit images

Output: grid of integers

Sizes: 4 x 4, 6 x 6, 9 x 9

Model:

$A, b \rightarrow$ input-independent, learnable

$c = \mathrm{CONCAT}(c_{0,0}, \ldots, c_{H,W})$

$c_{i,j} = \mathrm{LeNet}(x_{i,j})$

Equality (only for 9x9):

$Ay + b = 0 \rightarrow Ay + b \geq -\epsilon, Ay + b \leq \epsilon$

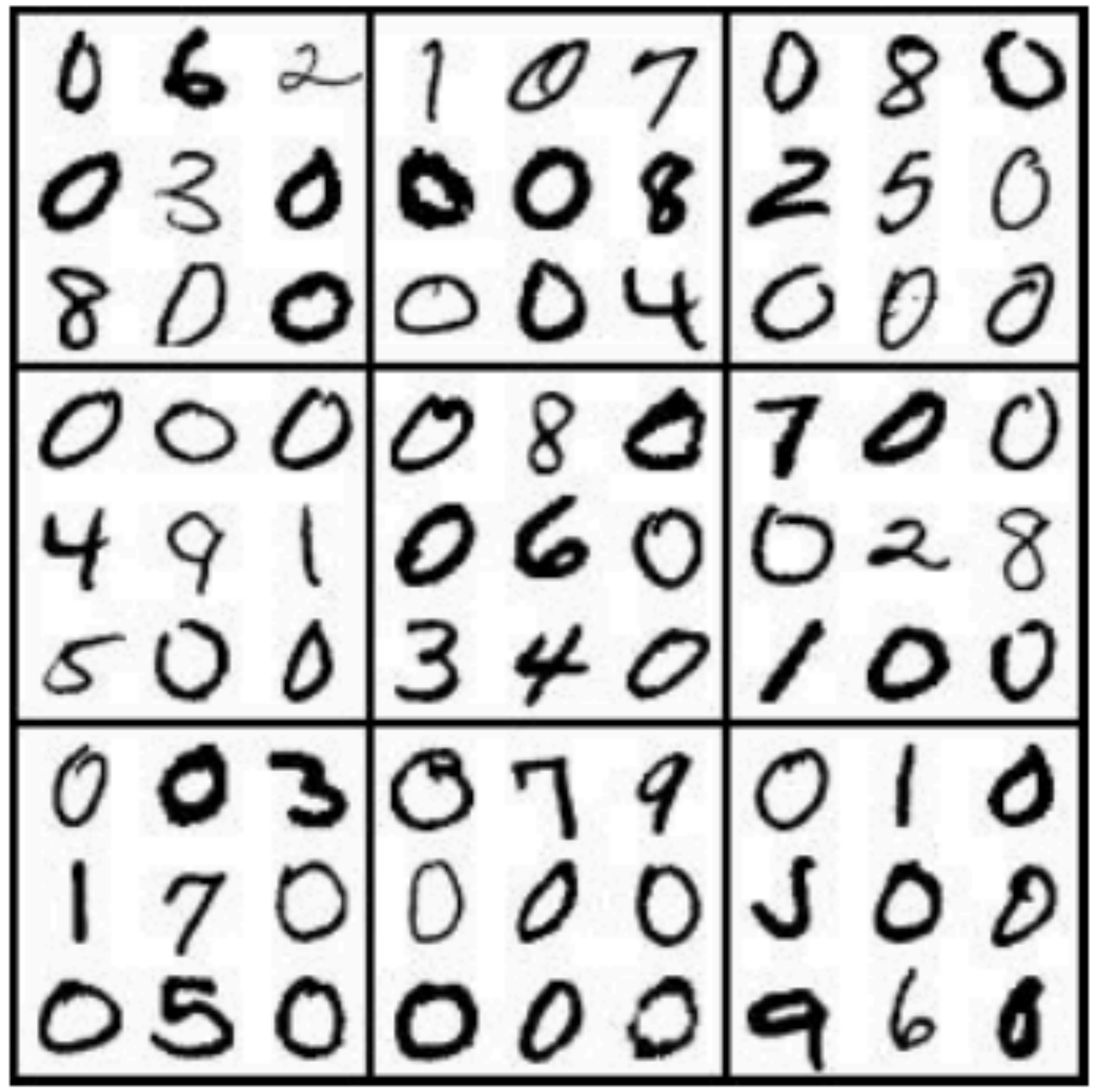


Figure 3. An example visual Sudoku image input, i.e. an image of a Sudoku board constructed with MNIST digits. Cells filled with the numbers 1-9 are fixed, and zeros represent unknowns.

Figure from *Wang, Po-Wei et al. "SATNet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver." ICML (2019).*

Visual Sudoku: Results

Table 1: Board accuracy and training time for different board sizes of visual sudoku (TO refers to time-out of 12 hours)

	Board Accuracy (in %)			Training Time (in min.)		
	4 x 4	**6 x 6**	**9 x 9**	**4 x 4**	**6 x 6**	**9 x 9**
Neural (RRN)	99.4	**97.6**	74.4	120	65	97
CombOptNet	0.0	0.0	0.0	TO	TO	TO
ILP–Loss (Ours)	**99.8**	94.7	**99.0**	**2**	**5**	**45**

* For 6 x 6 sudoku, 4.4% of the queries timed out with our learned constraints

Random Constraints: Setup

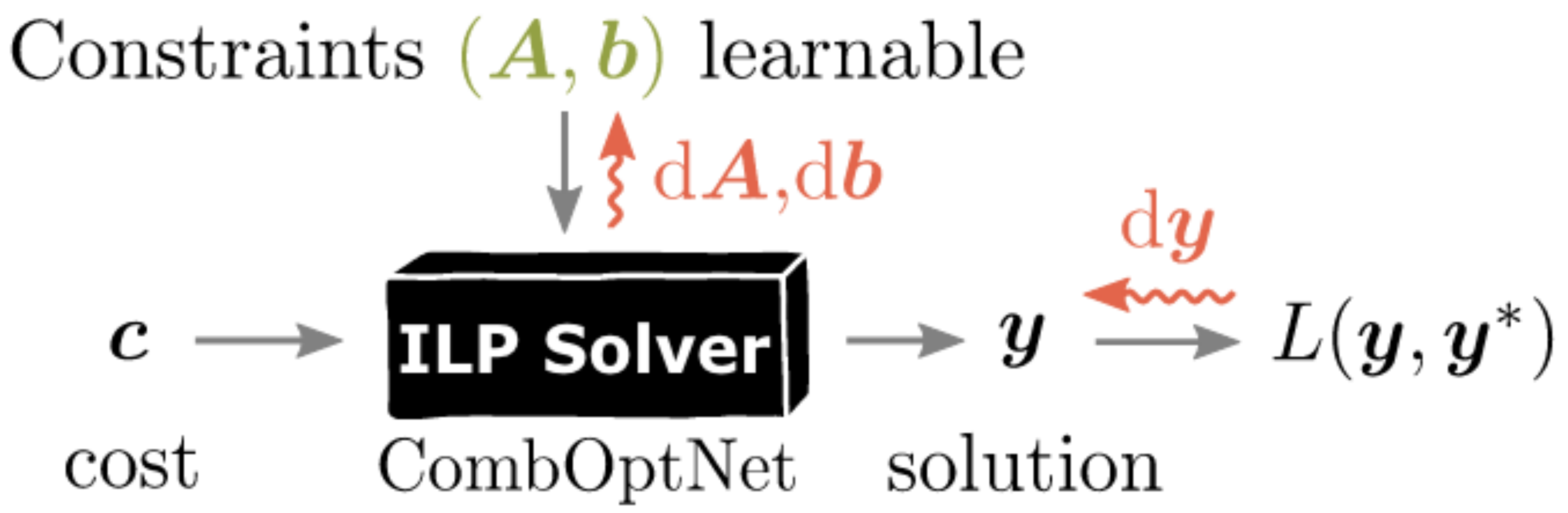


Figure from *Paulus, Anselm et al. "CombOptNet: Fit the Right NP-Hard Problem by Learning Integer Programming Constraints." ICML (2021).*

Domain: binary [0, 1], dense [-5, 5]

Constraint matrix: 1 x 16, 2 x 16, 4 x 16, 5 x 16 (ground truth)

Learnable constraints: 2 x #(ground truth constraints)

Dataset seeds: 10

Random Constraints: Results

Table 2: Mean $\pm$ std err of the vector accuracy ($\mathbf{M}_\Theta(x) = \mathbf{y}^*$) and training time over the 10 problem instances in each setting. Number of learnable constraints is twice the number of ground truth constraints.

	Binary				**Dense**			
	1	**2**	**4**	**8**	**1**	**2**	**4**	**8**
	Vector Accuracy (mean $\pm$ std err in %)							
CombOptNet	97.0 ± 1.0	96.4 ± 0.3	89.5 ± 1.6	61.2 ± 3.9	85.9 ± 1.0	71.2 ± 2.3	29.6 ± 2.8	2.7 ± 1.0
ILP–Loss (Ours)	$\mathbf{97.8 \pm 0.4}$	$\mathbf{96.4 \pm 0.5}$	$\mathbf{93.5 \pm 0.8}$	$\mathbf{88.6 \pm 3.6}$	$\mathbf{96.6 \pm 0.3}$	$\mathbf{86.3 \pm 2.3}$	$\mathbf{74.0 \pm 5.4}$	$\mathbf{41.5 \pm 5.7}$
	Training Time (mean $\pm$ std err in minutes)							
CombOptNet	44.5 ± 0.5	43.4 ± 0.5	48.7 ± 1.1	56.5 ± 3.2	38.6 ± 1.5	40.9 ± 0.1	39.6 ± 0.7	61.4 ± 1.6
ILP–Loss (Ours)	$\mathbf{6.9 \pm 2.3}$	$\mathbf{15.4 \pm 2.3}$	$\mathbf{28.9 \pm 4.8}$	$\mathbf{42.0 \pm 7.2}$	$\mathbf{12.9 \pm 2.0}$	$\mathbf{27.2 \pm 3.7}$	$\mathbf{30.3 \pm 4.4}$	$\mathbf{35.2 \pm 7.6}$

Keypoint Matching: Setup

Fig. 1: Example keypoint matchings of the proposed architecture on SPair-71k.

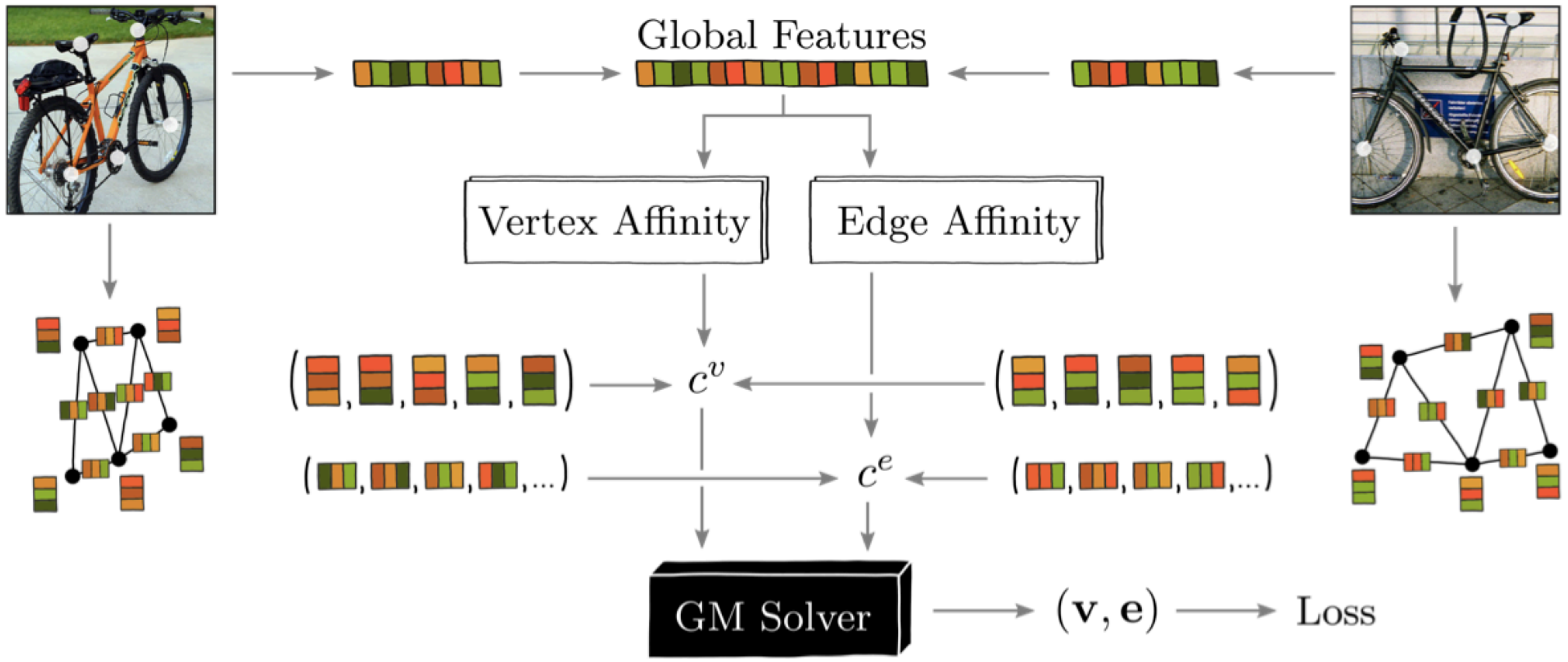


Fig. 5: Construction of combinatorial instance for keypoint matching.

Figure from *Rol'inek, Michal et al. "Deep Graph Matching via Blackbox Differentiation of Combinatorial Solvers." ECCV (2020).*

- **Dataset sizes:** 4, 5, 6, 7 (#keypoints)

Keypoint Matching: Results

Table 3: Pointwise accuracy for matching keypoints in two images for 4 datasets with varying number of keypoints. Neural+CI is the ILP inference with the known constraints over the cost learnt by neural model.

	4	**5**	**6**	**7**	**Avg.**
Neural	81.98	78.93	76.06	74.68	77.91
Neural + CI	83.32	80.28	77.68	76.66	79.49
CombOptNet	**83.63**	80.91	77.29	76.28	79.53
ILP–Loss (Ours)	83.50	**80.98**	**78.62**	**76.80**	**79.98**

Technique

Conversion to Constraint Satisfaction

Original ILP: $\arg\min_{\mathbf{z}} \mathbf{c}^T\mathbf{z}$ subject to $\mathbf{A}\mathbf{z} + \mathbf{b} \geq \mathbf{0}, \mathbf{z} \in \mathbb{Z}^n$

Modified ILP: $\arg\min_{\mathbf{z}} \mathbf{0}^T\mathbf{z}$ subject to $\mathbf{A}\mathbf{z} + \mathbf{b} \geq \mathbf{0}$; $\mathbf{c}^T\mathbf{z} \leq \mathbf{c}^T\mathbf{y}^*$; $\mathbf{z} \in \mathbb{Z}^n$

Target solution is the unique integral point feasible for the modified ILP.

Constraint Satisfaction Framework

- Geometry & ML perspective:

  contraint $\leftrightarrow$ hyperplane $\leftrightarrow$ linear classifier

  all constraints $\leftrightarrow$ polytope $\leftrightarrow$ polytope classifier

- Supervision dichotomy:

  Inference $\Longrightarrow$ combinatorial search via ILP solver

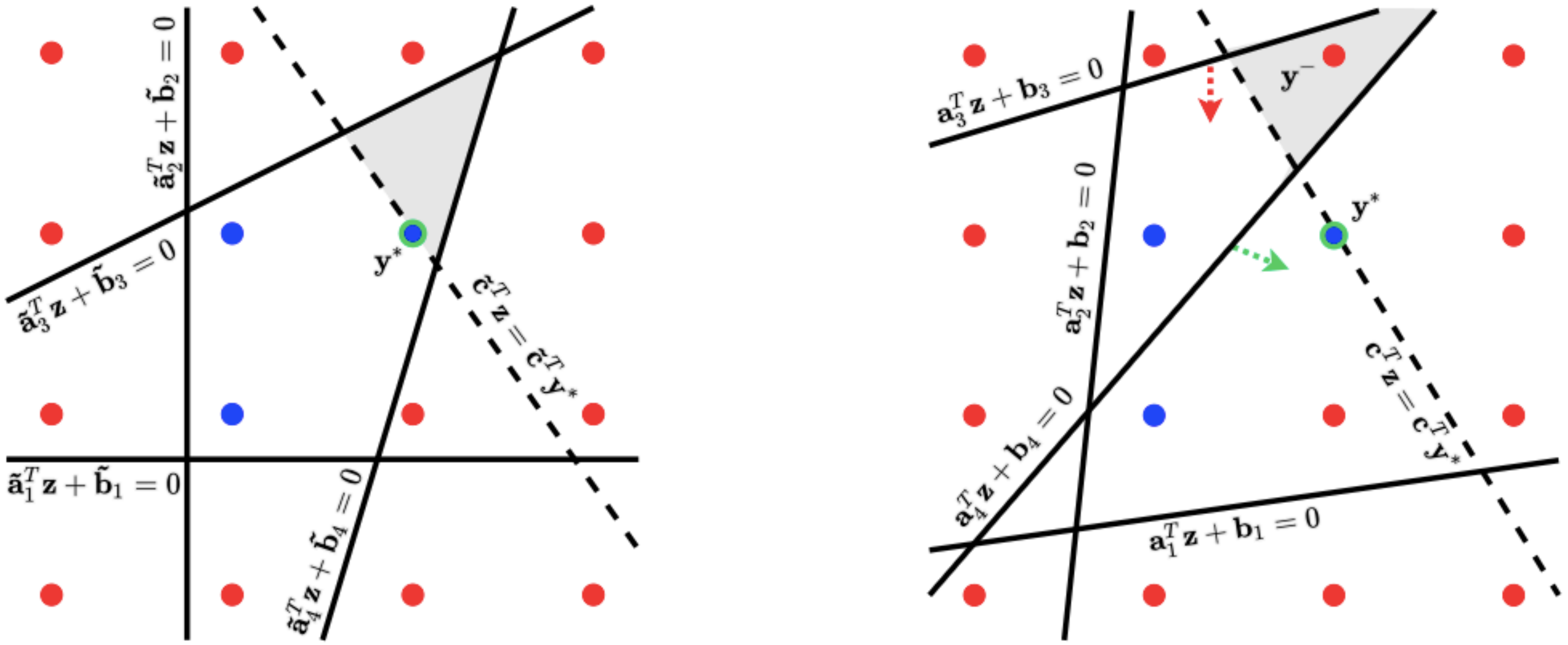
  Training $\Longrightarrow$ binary classification on +ve & -ve examples

- Example source:

  +ve example $\leftarrow$ unique, target solution

  -ve example(s) $\leftarrow$ exponential, sampling strategy?

Illustration



(a) Ground Truth ILP

(b) Intermediate Learnt ILP

Figure 1: An illustration of our framework. Figure on the left shows 4 ground truth constraints that need to be learnt. Blue dots are the only feasible integral points w.r.t. the 4 constraints. Shaded area containing only the dot with green border is the feasible region after we add the cost constraint (dashed line). Figure on the right shows an intermediate scenario while learning. The green-bordered dot (positive) is outside the intermediate 4^{th} constraint and the red dot (negative) is inside the intermediate 3^{rd} constraint. Positive and negative losses encourage the 4^{th} and the 3^{rd} hyperplanes to move in the direction shown by the green and red dotted arrows respectively.

Solver-free ILP Loss

$$L(\mathbf{A}, \mathbf{b}, \mathbf{c}, \mathbf{y}^* | \mathcal{N}_{\mathbf{y}^*}) = \lambda_{pos} L_+ + \lambda_{neg} L_- + \lambda_{cov} L_o \quad \text{where} \tag{5}$$

$$L_+ = \frac{1}{m} \sum_{i=1}^{m} \max\{0, \mu^+ - d(\mathbf{y}^*; [\mathbf{a}_i | \mathbf{b}_i])\} \tag{6}$$

$$L_- = \frac{1}{|\mathcal{N}_{\mathbf{y}^*}|} \sum_{\mathbf{y}^- \in \mathcal{N}_{\mathbf{y}^*}} \sum_{i=1}^{m+1} w^i_{\mathbf{y}^-} \max\{0, \mu^- + d(\mathbf{y}^-; [\mathbf{a}_i | \mathbf{b}_i])\} \tag{7}$$

$$L_o = \sum_{i,j=1; i \neq j}^{m} \frac{\mathbf{a}_i^T \mathbf{a}_j}{|\mathbf{a}_i||\mathbf{a}_j|} \simeq \left(\sum_{i=1}^{m} \frac{\mathbf{a}_i}{|\mathbf{a}_i|} \right)^2 \tag{8}$$

$$w^i_{\mathbf{y}^-} = \frac{e^{\left(-d\left(\mathbf{y}^-; [\mathbf{a}_i | \mathbf{b}_i]\right)/\tau\right)}}{\sum_{j=1}^{m+1} e^{(-d(\mathbf{y}^-; [\mathbf{a}_j | \mathbf{b}_j])/\tau)}} \tag{9}$$

Temperature Annealing

- Assignment of -ve to hyperplane
- **High temperature** (initially)
  $\Longrightarrow$ soft assignment to multiple hyperplanes
  $\Longrightarrow$ exploration
- **Low temperature** (finally)
  $\Longrightarrow$ hard assignment to best hyperplane
  $\Longrightarrow$ exploitation
- Encourages multiple constraint violations per -ve $\Longrightarrow$ robust constraints

Negative Samplers

- **K-hop neighbors (L1 dist = K) for small K:**
  - hardest -ves

- **Project on hyperplane and round to an integral neighbor:**
  - takes training progress into account
  - easiest misclassified -ves
  - signal to each hyperplane (roughly)

- **Other target solutions in the batch:**
  - signal to cost parameters (for input-independent original constraints)

Extra

What didn't work

- Max-margin formulation as in SVM
  - learnability issues
- High value of margin (we use 0.01 in our experiments)
  - harder to optimize
- Cross-entropy based replacement for hinge loss
  - instability; loss terms non-zero even without mis-classification
- Lagrangian weighting for loss terms
  - too sensitive to the new hyperparams

Future Directions (1/3)

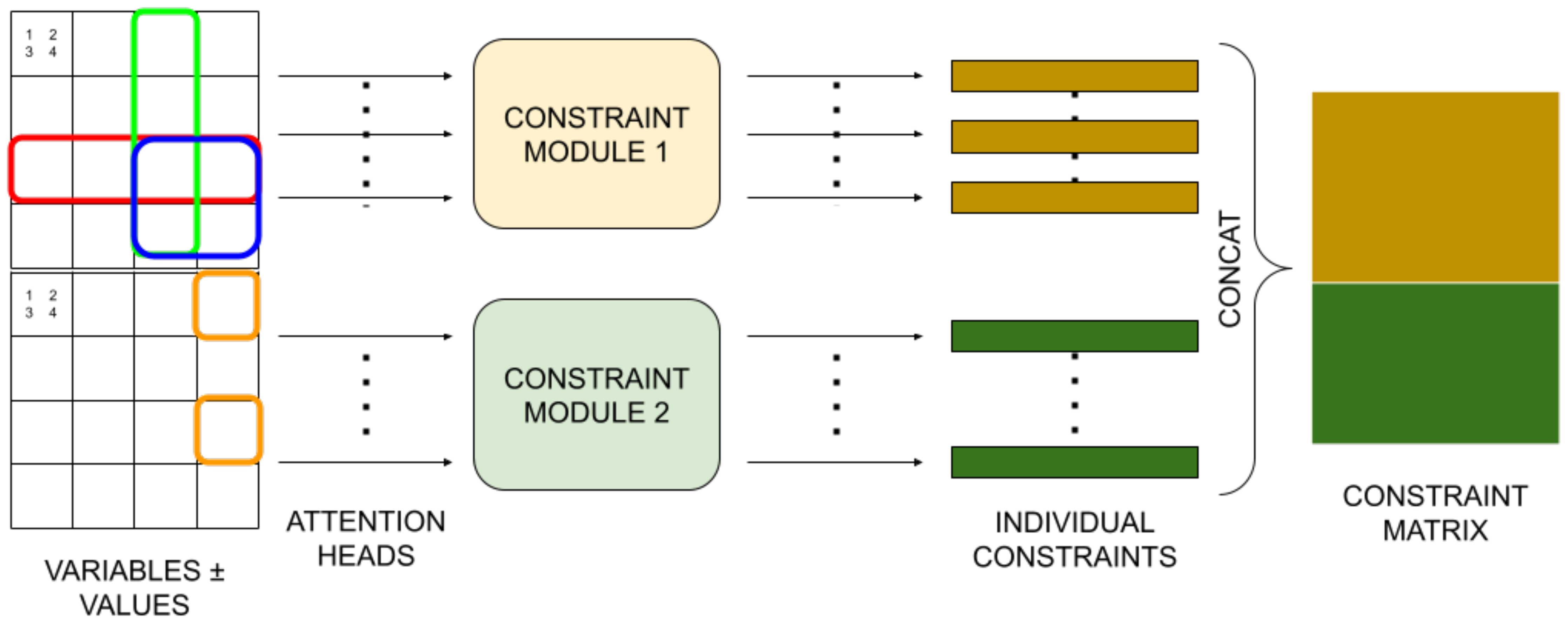
1. **Auxiliary variables:**
   - strictly richer modelling capacity
   - supervision available only for primary variables
   - idea: "$\exists$ valuation of aux vars for which +ve is inside polytope",
     " $\nexists$ valuation of aux vars for which -ve is inside polytope"

2. **Mixed Integer Linear Programs (special case: Linear Programs):**
   - extends to continuous domains
   - idea: "+ve lies on boundary in the continuous variable dimensions"

Future Directions (2/3)

3. Constraint parametrization:

- priors such as lifted constraints
- scale-independent parametrization (eg. same model for 4 x 4, 6 x 6, 9 x 9 sudoku)
- faster inference by aligning with solver optimizations
- sample efficiency / better learnability



Future Directions (3/3)

4. **Extension to Neural-Symbolic-Neural architectures**
   - idea: "maintain an estimate of the intermediate target solution, updated via gradient descent and kept integral via quantization-like mechanisms"

5. **Distill learnt constraints:**
   - into human-interpretable form, or,
   - into faster-to-solve form

Thank You!

Questions?