MAX PLANCK INSTITUTE
FOR INTELLIGENT SYSTEMS

CombOptNet: Fit the Right NP-Hard Problem by Learning Integer Programming Constraints

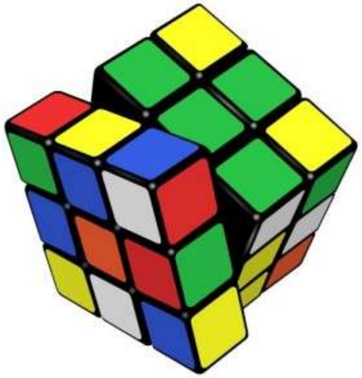
A. Paulus
M. Rolínek
V. Musil
B. Amos
G. Martius



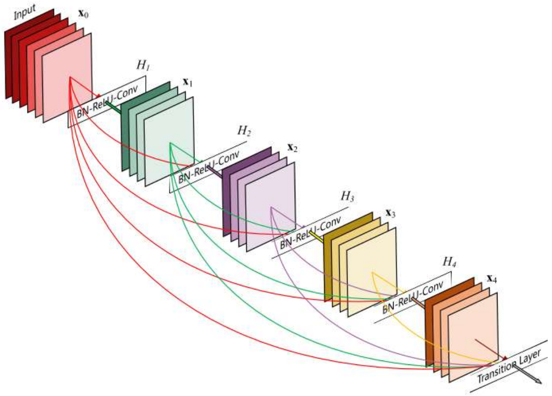


Motivation

Combinatorial optimization

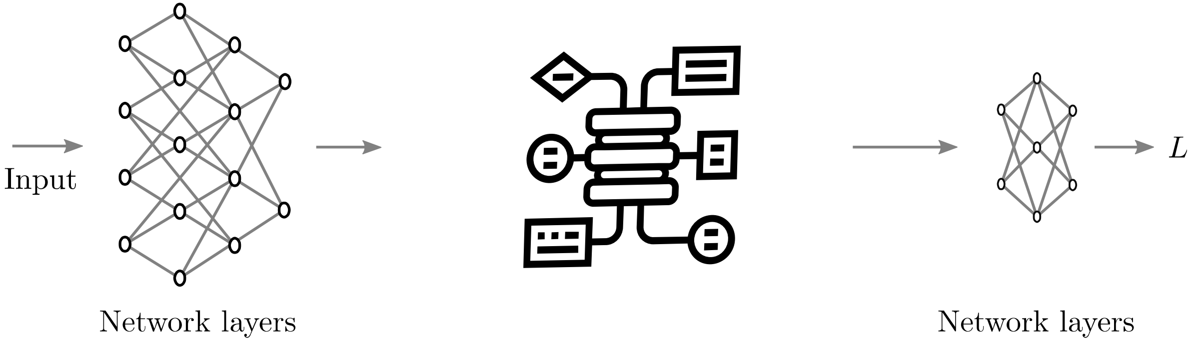


Deep Learning



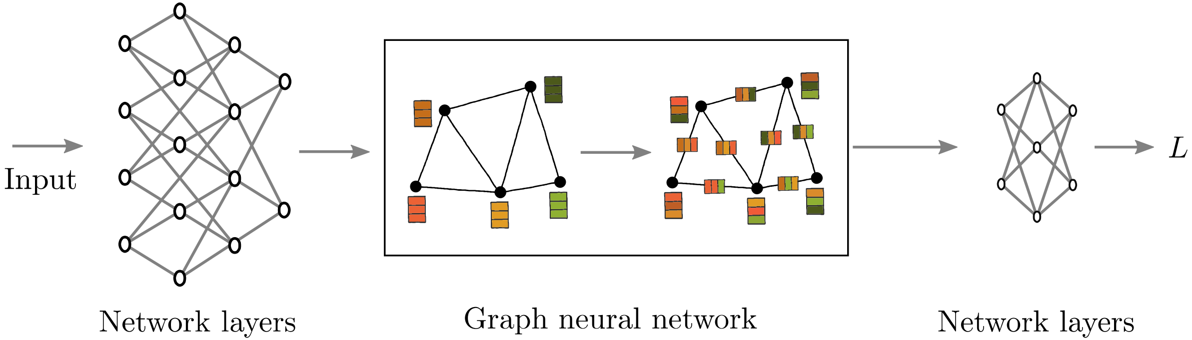
Goal

- Include combinatorial block in architecture



Approaches

- GNNs: Mimic combinatorial solver for graph problems with message passing [1]



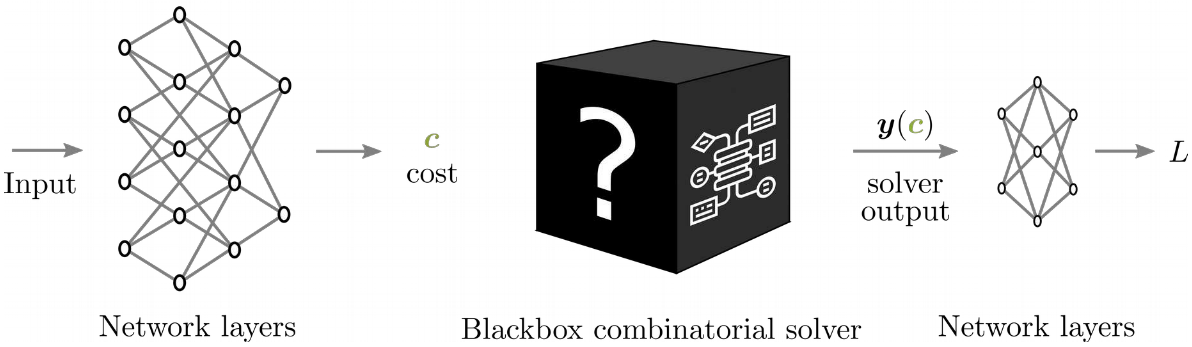
- Drawback: Suboptimal from combinatorial perspective

Approaches

- **Softened solver**: Design differentiable softened versions of solvers [2, 3]

Approaches

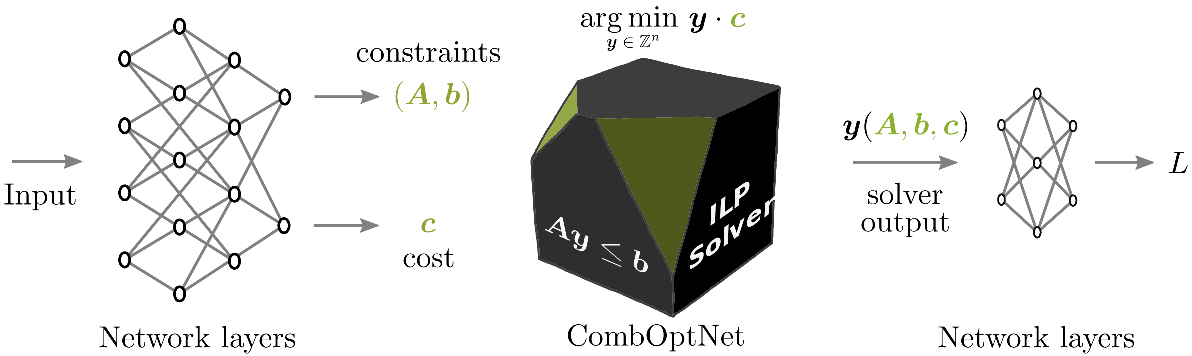
- Blackbox Differentiation: Blackbox backprop [4, 5, 6], differentiable perturbed optimizers [7]



- Drawback: Commit to specific combinatorial problem

Our Approach

- CombOptNet: ILP solver as differentiable layer

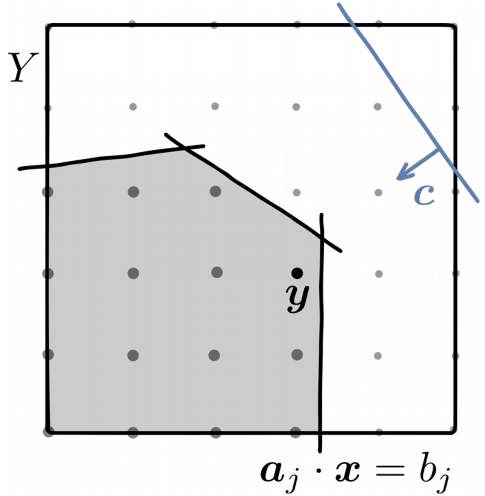


Integer Linear Programs

ILP

$$y(A, b, c) = \arg\min_{y \in Y} \; y \cdot c$$
$$\text{subject to } Ay \le b$$

Y .... bounded subset of $\mathbb{Z}^n$
$c \in \mathbb{R}^n$ .... cost
$A = [a_1, \ldots, a_k] \in \mathbb{R}^{k \times n}$... constraints
$b \in \mathbb{R}^k$ .... bias term



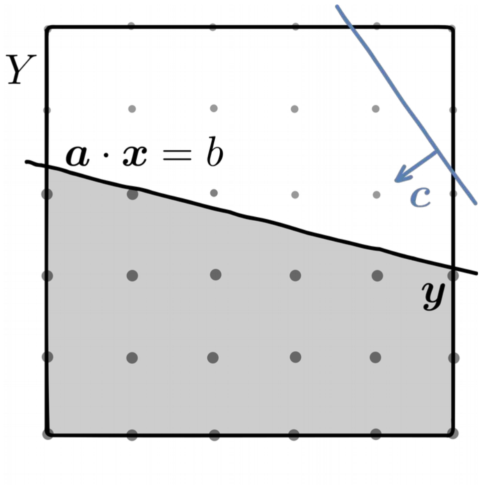
Integer Linear Programs

Knapsack as ILP

$$\boldsymbol{y}(\boldsymbol{a}, b, \boldsymbol{c}) = \arg\min_{\boldsymbol{y} \in Y} \boldsymbol{y} \cdot \boldsymbol{c}$$

$$\text{subject to } \boldsymbol{a} \cdot \boldsymbol{y} \leq b$$

$Y = \{0,1\}^n$ .... binary hypercube
$\boldsymbol{c} \in \mathbb{R}^n$................. neg. item prices
$\boldsymbol{a} = [a_1, \ldots, a_n] \in \mathbb{R}^n$... item weights
$b \in \mathbb{R}$.................. knapsack capacity



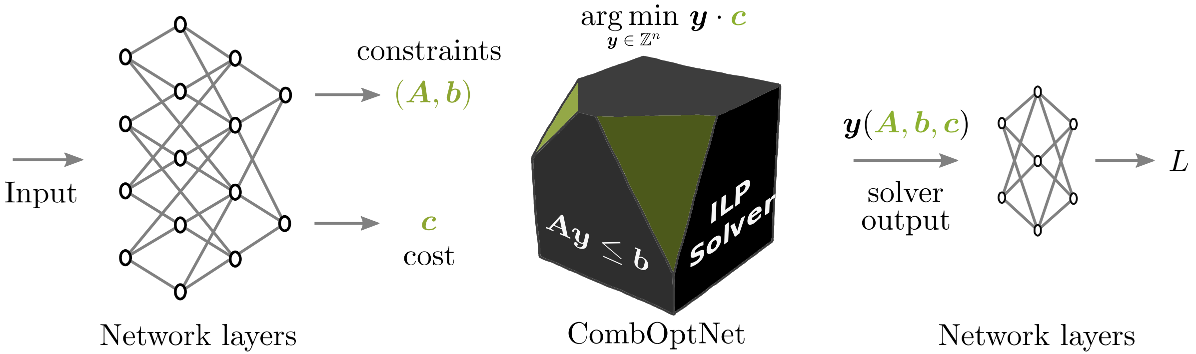
- Express NP-hard problems as ILPs
- Learning constraints = learning combinatorial nature

ILP

$$y(A, b, c) = \arg\min_{y \in Y} \; y \cdot c$$

$$\text{subject to } Ay \leq b$$

Architecture



- Generality: Aspires universal combinatorial expressivity
- Competitive combinatorics: SOTA ILP solver (Gurobi [8])

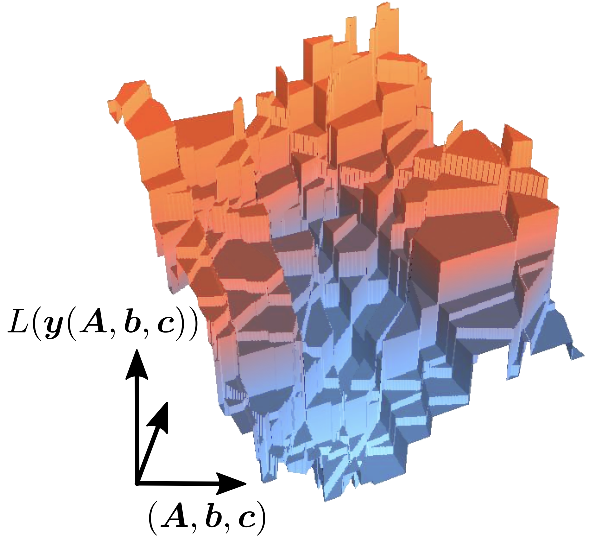
How? — The Difficulty

ILP

$$\boldsymbol{y}(\boldsymbol{A},\boldsymbol{b},\boldsymbol{c}) = \arg\min_{\boldsymbol{y}\in Y} \boldsymbol{y}\cdot\boldsymbol{c}$$

$$\text{subject to } \boldsymbol{A}\boldsymbol{y} \leq \boldsymbol{b}$$

- Small perturbation typically does not change solution
- True gradient exists but is uninformative
- Cost:
  - Well-studied [4, 7, 9, 10, 11]
- Constraints:
  - Mostly unexplored
  - Difficult (no active constraints)



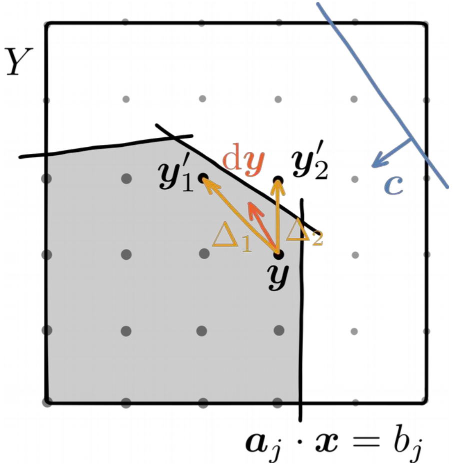
How? — Components

ILP

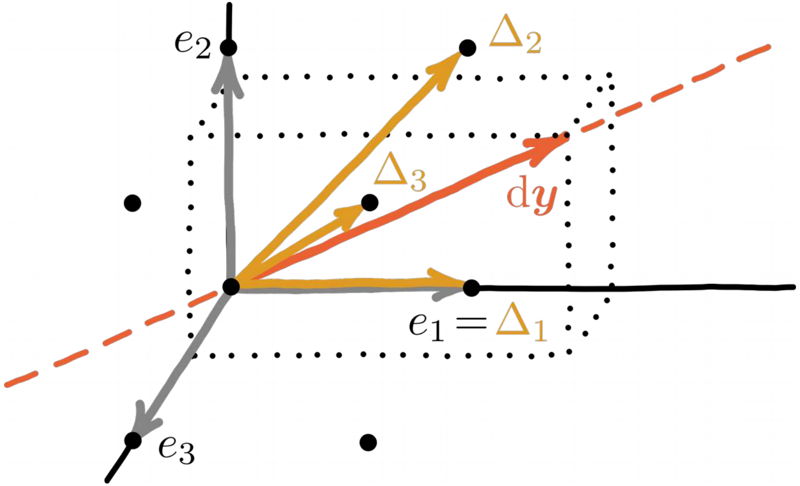
$$\boldsymbol{y}(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}) = \arg\min_{\boldsymbol{y} \in Y} \boldsymbol{y} \cdot \boldsymbol{c}$$

subject to $\boldsymbol{A}\boldsymbol{y} \leq \boldsymbol{b}$

- Decomposition of incoming gradient into integer basis
    - 2D



    - 3D



- Attainable update targets

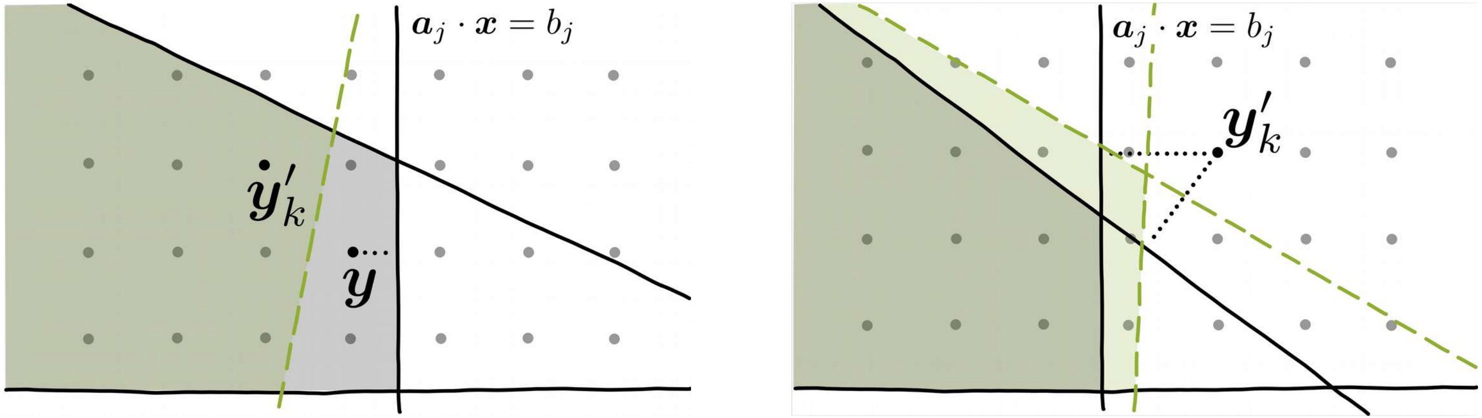
How? — Components

ILP

$$y(A, b, c) = \arg\min_{y \in Y} y \cdot c$$

subject to $Ay \leq b$

- Gradient of geometry-aware mismatch function



- Generalizes concept of active constraints
- Precise definitions & algorithm on poster

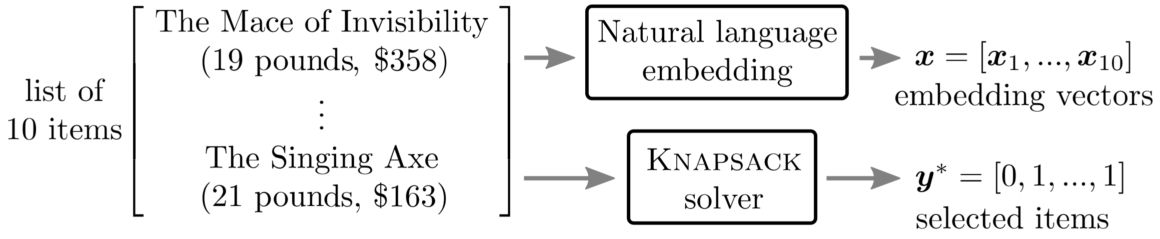
Experiments — Knapsack

ILP

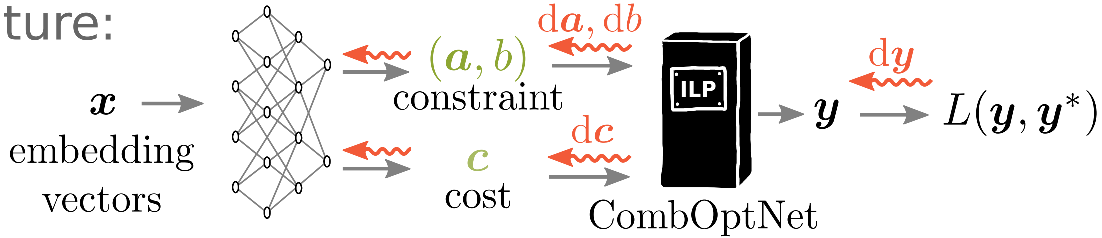
$$y(A, b, c) = \arg\min_{y \in Y} y \cdot c$$

subject to $Ay \leq b$

- Dataset:

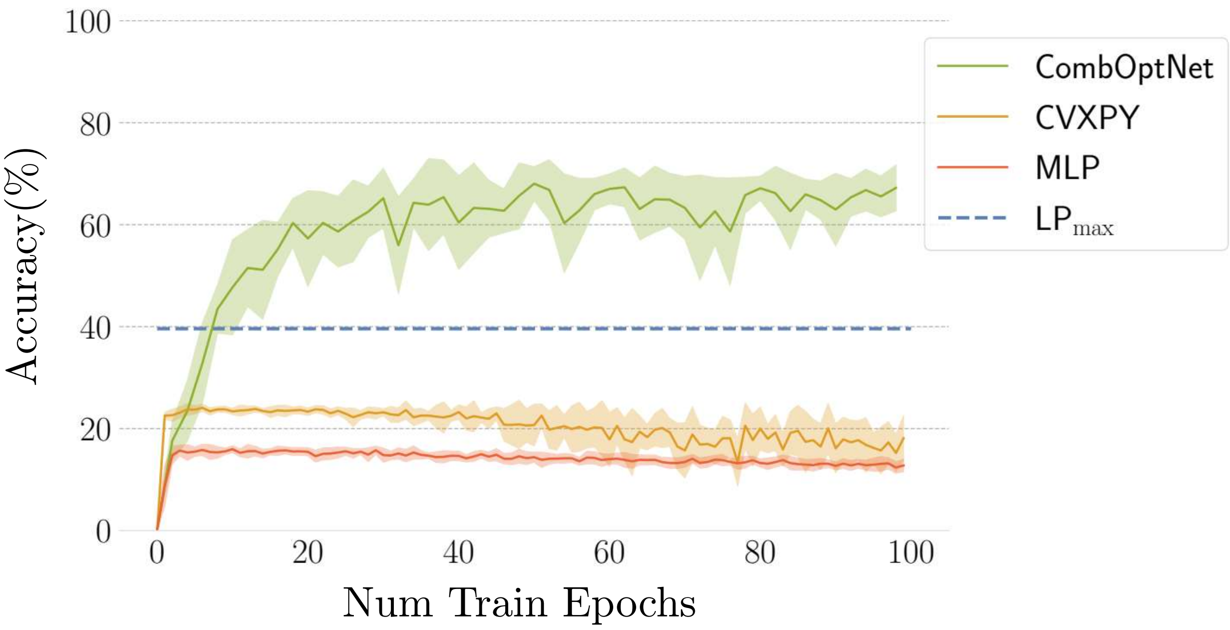


- Architecture:



Experiments — Knapsack

- Results:



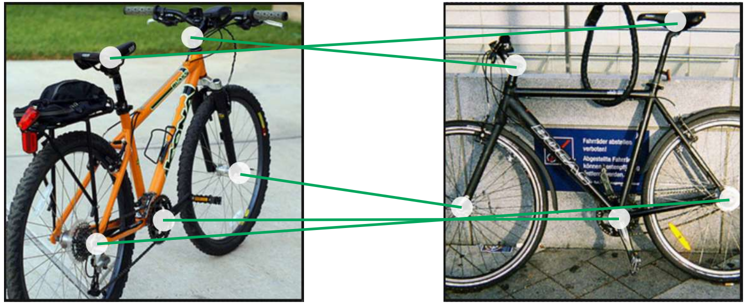
Experiments — Keypoint Matching

ILP

$$y(A, b, c) = \arg\min_{y \in Y} y \cdot c$$

subject to $Ay \leq b$

- Dataset:



- Architecture: Blackbox Graph Matching [6]

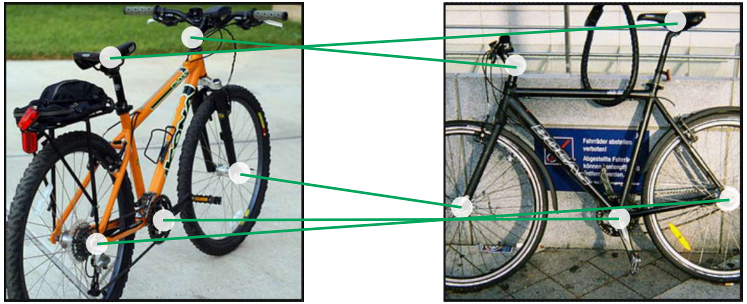
Experiments — Keypoint Matching

ILP

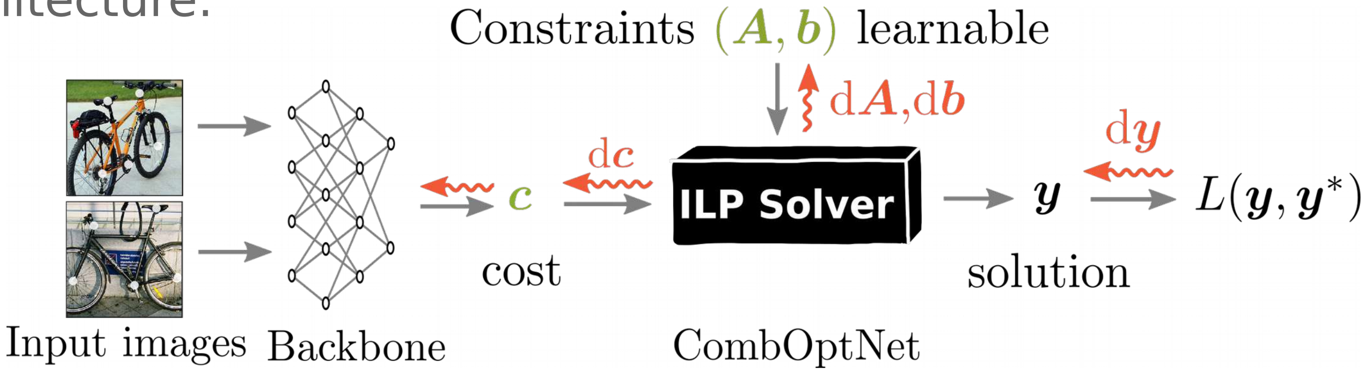
$$y(A, b, c) = \arg\min_{y \in Y} y \cdot c$$

$$\text{subject to } Ay \leq b$$

- Dataset:



- Architecture:

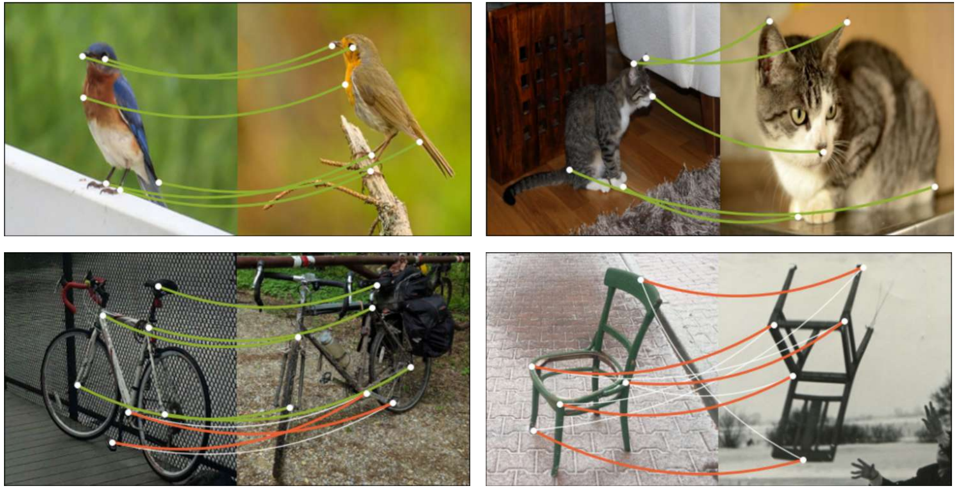


Experiments — Keypoint Matching

- Results:

Method	4×4	5×5	6×6	7×7
CombOptNet	83.1	80.7	78.6	76.1
BB-GM	84.3	82.9	80.5	79.8

- Examples:



References

github.com/martius-lab/CombOptNet

- [1] Q. Cappart, D. Chételat, E. Khalil, A. Lodi, C. Morris, P. Veličković
  Combinatorial optimization and reasoning with graph neural networks
- [2] M. Fey, J. Lenssen, C. Morris, J. Masci, N. Kriege
  Deep Graph Matching Consensus, ICLR 2020
- [3] M. Engilberge, L. Chevallier, P. Pérez, M. Cord
  SoDeep: a Sorting Deep net to learn ranking loss surrogates, CVPR 2019
- [4] M. Vlastelica, A. Paulus, V. Musil, G. Martius, M. Rolínek
  Differentiation of Blackbox Combinatorial Solvers, ICLR 2020
- [5] M. Rolínek, V. Musil, A. Paulus, M. Vlastelica, C. Michaelis, G. Martius
  Optimizing Rank-Based Metrics with Blackbox Differentiation, CVPR 2020
- [6] M. Rolínek, P. Swoboda, D. Zietlow, V. Musil, A. Paulus, G. Martius
  Deep Graph Matching via Blackbox Differentiation, ECCV 2020
- [7] Q. Berthet, M. Blondel, O. Teboul, M. Cuturi, J. Vert, F. Bach
  Learning with Differentiable Perturbed Optimizers, NeurIPS 2020
- [8] Gurobi Optimization, 2019, http://www.gurobi.com
- [9] A. Ferber, B. Wilder, B. Dilkina, M. Tambe
  MIPaaL: Mixed Integer Program as a Layer, AAAI 2020
- [10] A. Elmachtoub, P. Grigas. Smart “Predict, then Optimize”
- [11] J. Mandi, E. Demirovic, P. Stuckey, T. Guns
  Smart Predict-and-Optimize for Hard Combinatorial Optimization Problems, AAAI 2020