VRIJE UNIVERSITEIT BRUSSEL

KU LEUVEN

Learning from environment in constraint solving

Tias Guns <*tias.guns@kuleuven.be*>

Joint work with:

- Jayanta Mandi
- Maxime Mulamba
- Victor Bucarey Lopez
- Rocs Canoy
- Ahmed K.A. Abdullah

And: (the last part)

- Peter Stuckey
- Emir Demirovic

THE UNIVERSITY OF MELBOURNE

- Michelangelo Diligenti
- Michele Lombardi

General outline

This session:

- ~60 min Principles of Data Science
- ~30 min Learning from user in CP
- ~30 min Practical (python notebook)

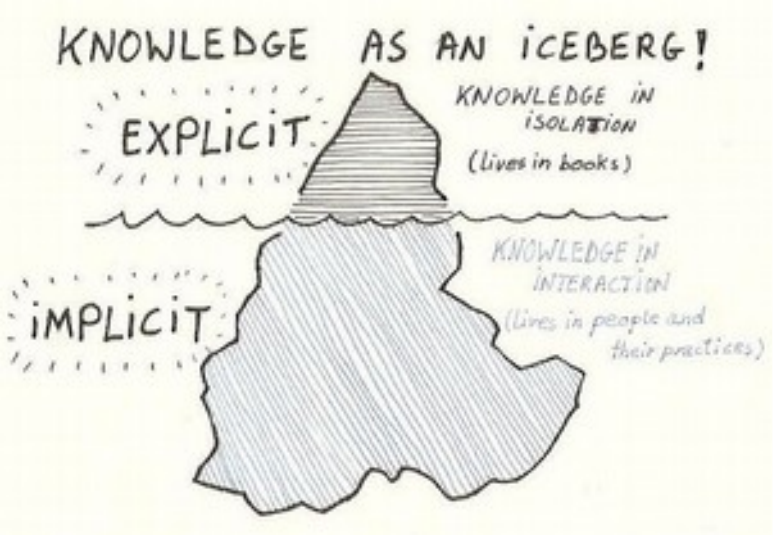
Afternoon session:

- ~20 min Learning from vision in CP
- ~40 min Learning from environment in CP
- ~30 min Practical (other python notebook)

Prediction + constraint solving

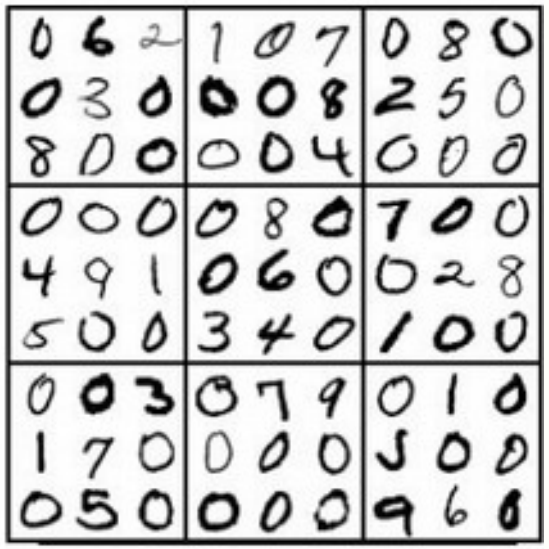
- Part <u>explicit</u> knowledge:
  in a formal language

- Part <u>implicit</u> knowledge:
  learned from data



  - tacit knowledge *(user preferences, social conventions)*
  - **perception *(vision, natural language, audio)***
  - **complex environment *(demand, prices, defects)***

Perception data and constraint solving



ML view:

Wang, Donti, Wilder, Kolter; ICML19

- can we learn the (pairwise) sudoku *constraints*?
- test limits of learning for reasoning

Our view part explicit, part implicit:

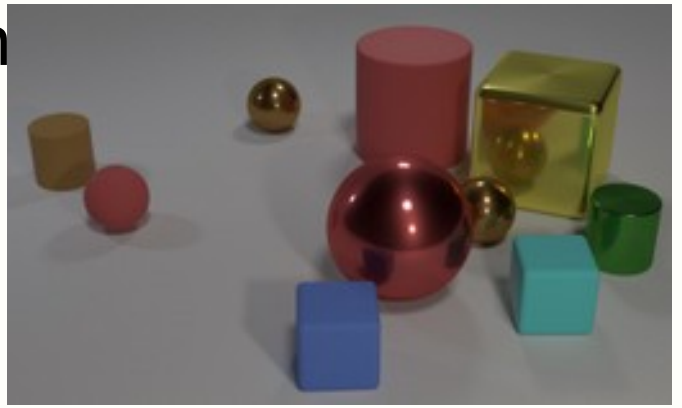
- know constraints, get predictions
- maximum likelihood problem?
- test limits of reasoning on learning

[Mulamba, Mandi, Canoy, Guns, CPAIOR20]

Perception data and constraint solving

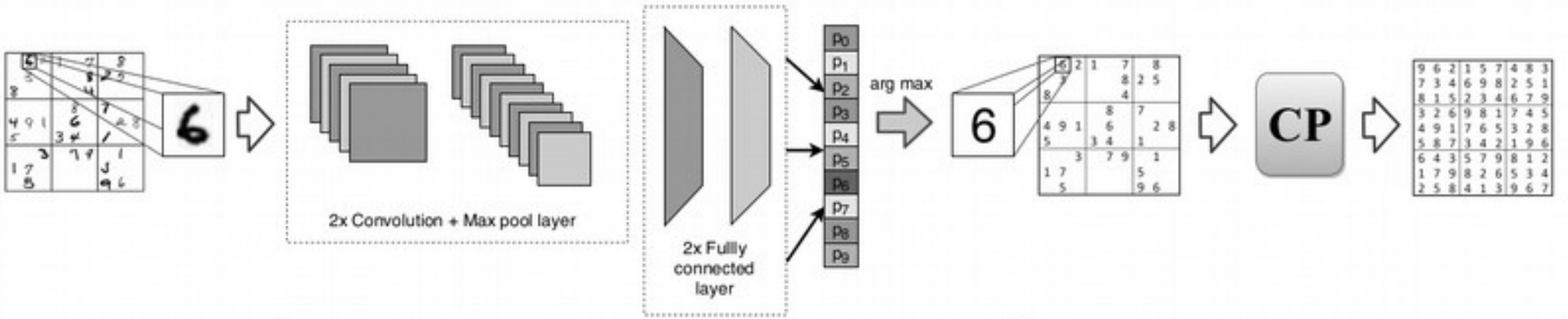
Other application settings:

- Document analysis
- Paper-based configuration problems (tax forms)
- Object-detection based reasonin
- ...



Perception-based constraint solving

Pedagogical instantiation: visual sudoku (naïve)



	accuracy			failure rate	time
	img	cell	grid	grid	average (s)
baseline	94.75%	15.51%	14.67%	84.43%	0.01

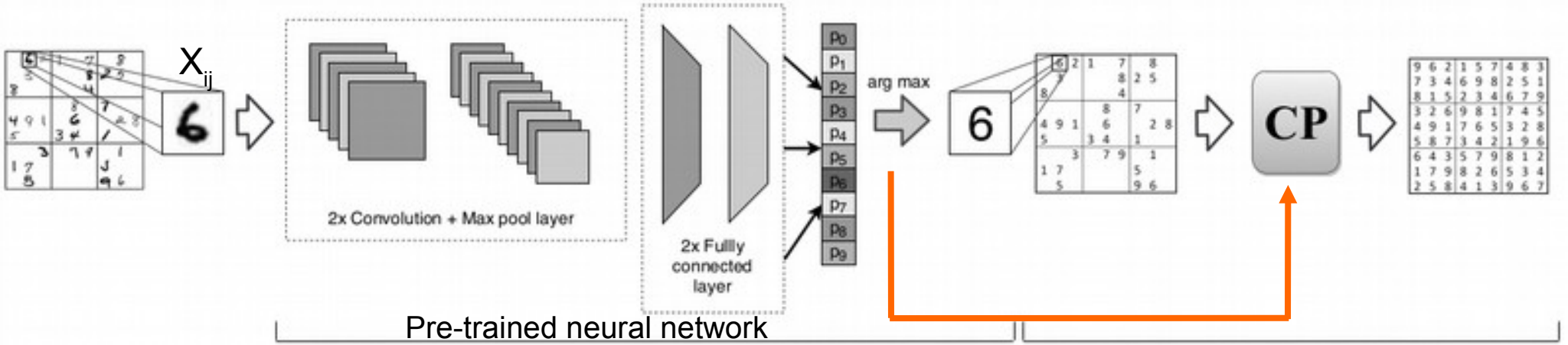
Perception-based constraint solving

- Inspired by visual sudoku in SATNet paper [Wang, Donti, Wilder, Kolter, ICML19]

- Q: Do we need *integrated* end-to-end learning?

  → Keep explicit explicit (constraints) and learn implicit (img class)

put more into the constraint solving



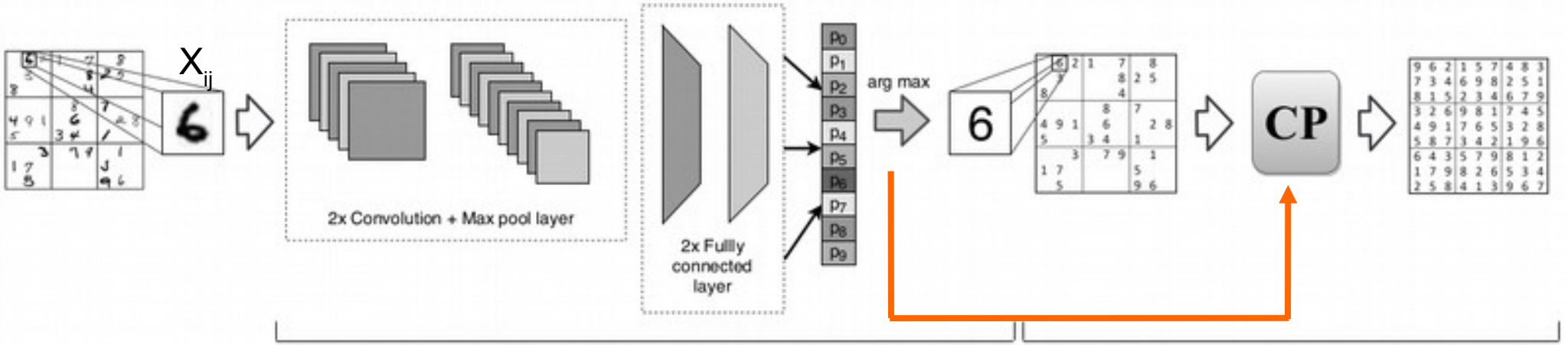
remove *arg max* of each individual prediction: $\hat{y}_{ij} = f_\theta(X_{ij}) = \underset{k \in \{0,..,9\}}{\arg\max} P_\theta(y_{ij} = k | X_{ij}),$

and make part of the CP model! $min \sum_{\substack{(i,j) \in \\ given}} \sum_{k \in \{1,..,9\}} \underbrace{-log(P_\theta(y_{ij} = k | X_{ij}))}_{constant} * \mathbb{1}[s_{ij} = k]$

s.t. sudoku(*s*)

= find the maximum likelihood solution!

put more into the constraint solving



What does that mean, maximum likelihood solution of a Sudoku?

The *unconstrained* max likelihood solution = argmax prediction of each image
→ sudoku constraint **forbids** certain solutions: find next most likely one...

In ML speak: the solver does 'joint inference' over *all* predictions, takes structure into account
ex. used in Natural Language Processing [Punyakanok, COLING04]

Back to the deep learning part

Why are individual predictions insufficient?

→ each image is learned independently, there is no penalty for '9's having very high likelihood of being an '8'

→ known as '**calibration**' in the ML community (overconfidence in some classes)

→ can actually distort the joint inference, but we can first calibrate the predictions!

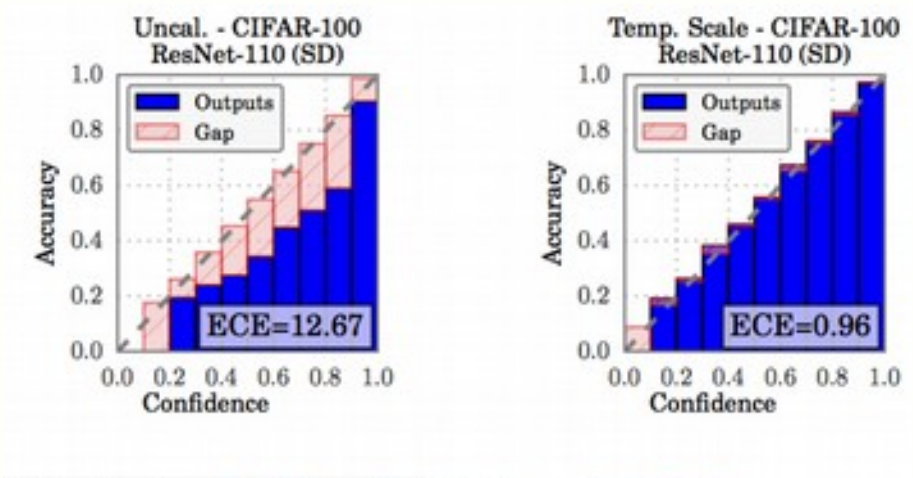


Illustration from Guo, Pleiss, Sun, Weinberger, ICML, 2017

	uncalibrated	Temperature scaling	Vector scaling	Matrix scaling
NLL	12.07%	11.61%	11.38%	**10.12%**
test acc.	96.75%	96.75%	96.70%	**96.93%**

Table 4: NLL loss (%) on validation set and accuracy on test set for Platt scaling variants

[Maxime Mulamba, Jayanta Mandi, Rocsildes Canoy, Tias Guns, CPAIOR20

Bacl to the constraint solving

Are we using all available information?

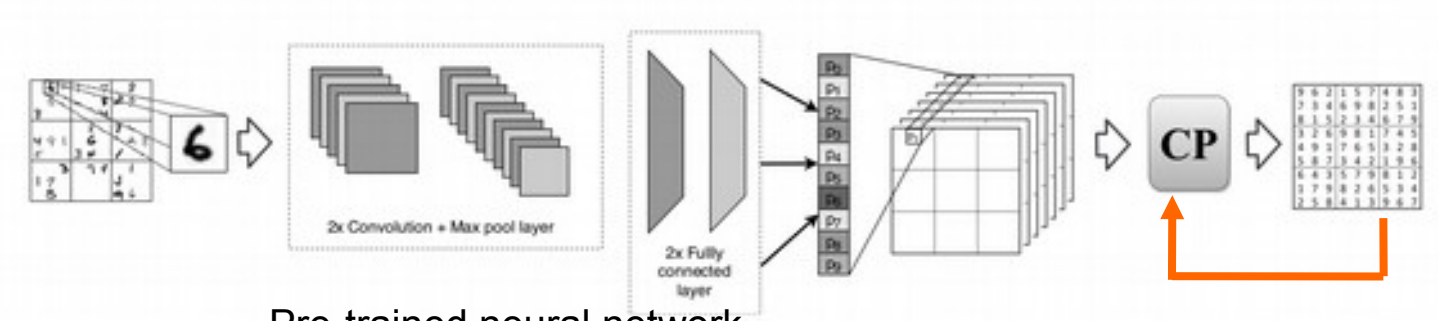
A sudoku puzzle has to have one unique solution

→ not in current constraint model: a 2nd order constraint

$$\begin{aligned} \underset{X}{\text{argmin}} \quad & f(X) \\ \text{subject to} \quad & C(X) \\ & \nexists X' : X \neq X', C(X') \end{aligned}$$

But we can check it!

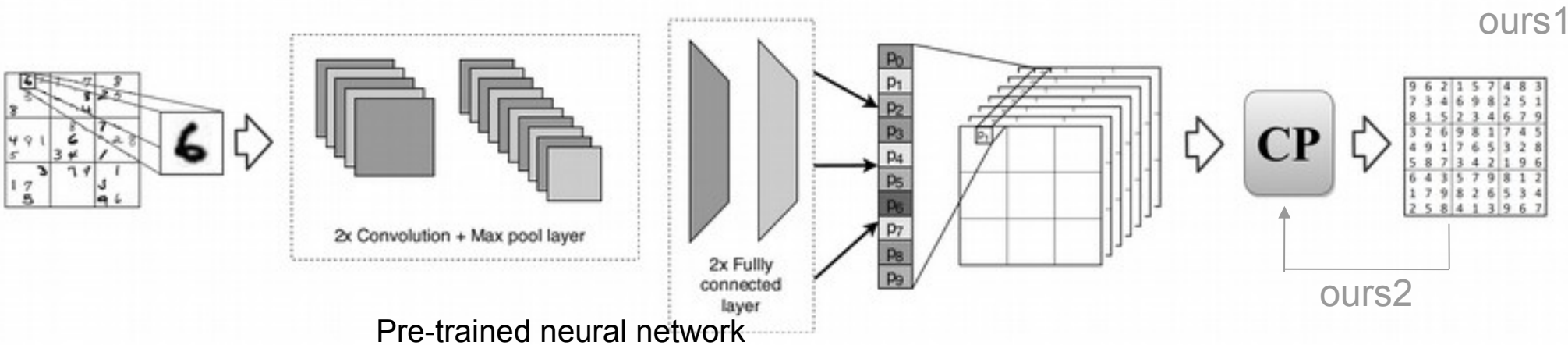
if the joint max likelihood image predictions has multiple solutions: **forbid** and find next most likely one!



Pre-trained neural network

Perception-based constraint solving

Hybrid: CP solver does *joint inference* over raw probabilities



Pre-trained neural network

	accuracy			failure rate	time
	img	cell	grid	grid	average (s)
baseline	94.75%	15.51%	14.67%	84.43%	0.01
hybrid1	99.69%	99.38%	92.33%	0%	0.79
hybrid2	99.72%	99.44%	92.93%	0%	0.83

[Maxime Mulamba, Jayanta Mandi, Rocsildes Canoy, Tias Guns, CPAIOR20]

Perception-based constraint solving

Practical later today:

you will implement and test hybrid approaches!

Enhancing the scope:

- Segmentation + classification + reasoning
- Object detection + segmentation + classification + reasoning

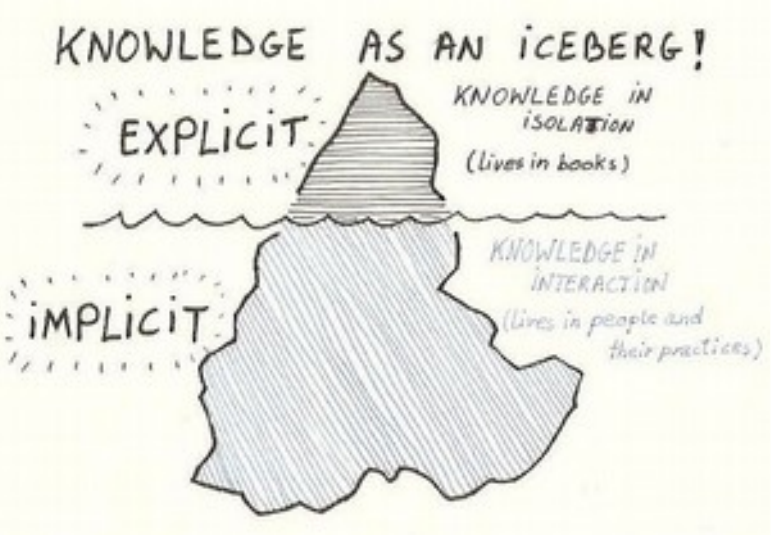
<DEMO>

http://178.62.207.46/home

Prediction + constraint solving

- Part <u>explicit</u> knowledge:
  in a formal language

- Part <u>implicit</u> knowledge:
  learned from data

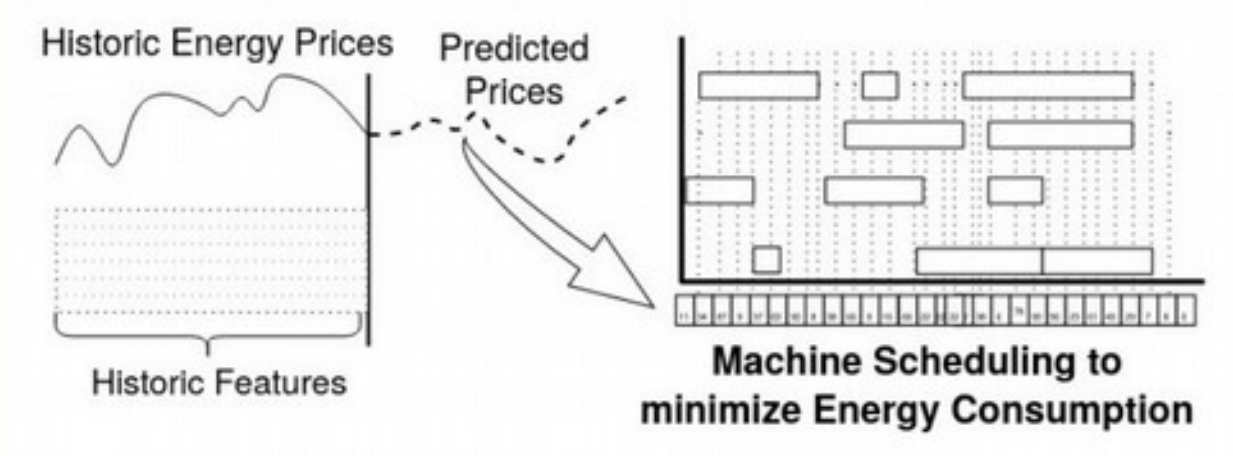


  - tacit knowledge *(user preferences, social conventions)*
  - perception *(vision, natural language, audio)*
  - **complex environment *(demand, prices, defects)***

Time for end-to-end learning!

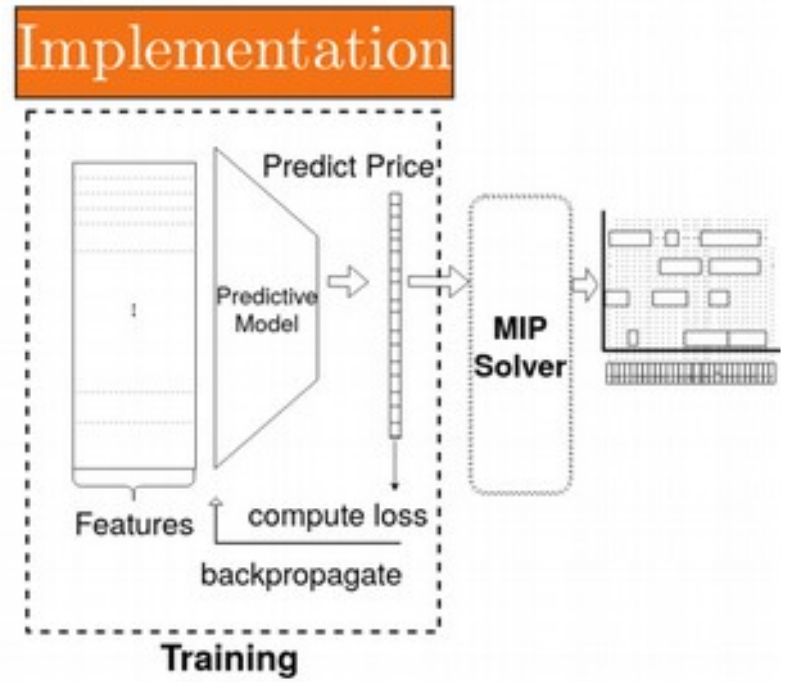
Complex environment (demand, prices)

Prediction + Optimisation aka decision-focussed learning:



- Optimize task scheduling's energy cost, by predicting energy prices
- Optimize steel plant manufacturing, by predicting steel defects
- Optimize money transport, by predicting amount of coins at clients

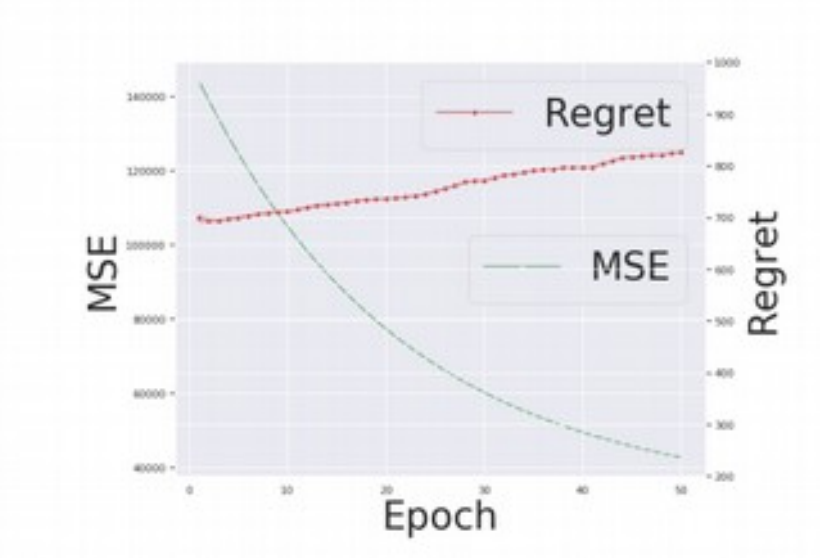
Prediction + Optimisation, naive



Pre-trained neural network

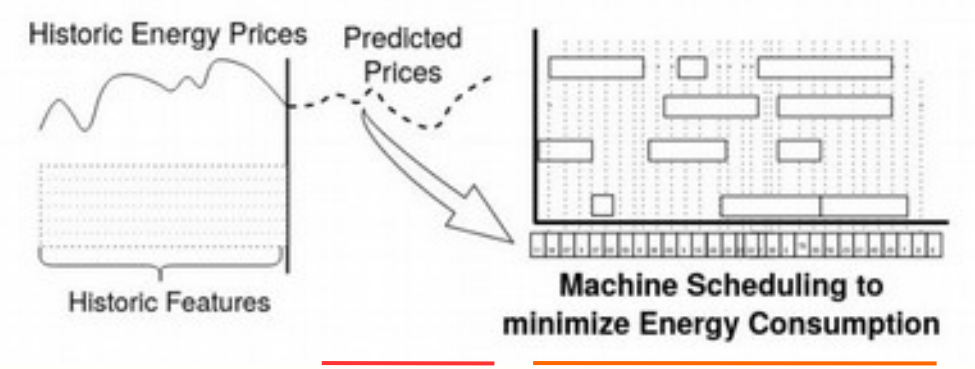
Can we do the (deep) learning better?

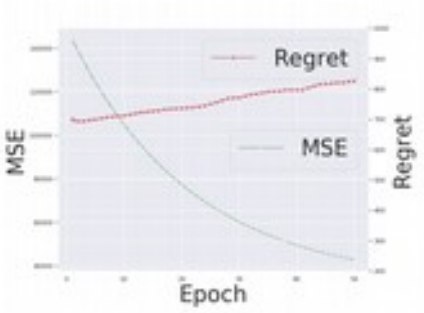
MSE loss function is not informative enough



MSE loss not the best proxy for *task* loss....

MSE loss not the best proxy for *task* loss....





Vector of predictions

Joint inference: trades off the individual predictions

<u>Why?</u>

- MSE = average of individual errors of the vector
- Joint inference = *joint* error

  → some errors worse than others!

Complex environment (demand, prices)

Which errors worse? is combinatorial, need to *solve* to know

Goal: end-to-end learning with *regret* as *loss*

$$regret(\{(\mathbf{x}, y)\}, f, \mathcal{V}, \mathcal{D}, \mathcal{C}, o) =$$
$$\sum_{j=1}^{B} o^{y_j}(V^{f_j}) - o^{y_j}(V^{y_j})$$
$$s.t.\ V^{f_j} \text{ solution of } COP(\mathcal{V}, \mathcal{D}, \mathcal{C}, o^{f_j})$$
$$V^{y_j} \text{ solution of } COP(\mathcal{V}, \mathcal{D}, \mathcal{C}, o^{y_j})$$

Challenges:

- *each* regret comp. is NP-hard
- over an exponential nr. of outcomes
- discrete & non-differentiable

[Smart Predict-and-Optimize for Hard Combinatorial Optimization Problems, Jayanta Mandi, Emir Demirovic, Peter Stuckey, Tias Guns. AAAI20]

Related work using deep learning (gradient descent)

Differentiable task losses for end-to-end learning:

Black box (subgradient methods):

→ - SPO+[1]: solve with f(2c* - c) (convex comb of real and predicted values)

- bb[2]: solve with f(c + eps) perturbed predictions

White box:

- QPTL[3]: solve Quadratic Program, differentiate KKT conditions
- Melding[4]: solve tightened LP relaxation as QP
- IntOpt[5]: solve LP with Interior Point, differentiate HSD

[1] Elmachtoub AN, Grigas P. Smart" predict, then optimize" arxiv, 2017
[2] Pogancic, Marin Vlastelica, et al. "Differentiation of Blackbox Combinatorial Solvers." ICLR. 2020
[3] Amos, Brandon, and J. Zico Kolter. "Optnet: Differentiable optimization as a layer in neural networks." ICML, 2017
[4] Wilder B, Dilkina B, Tambe M. “Melding the data-decisions pipeline: Decision-focused learning for comb. optimization.” AAA
[5] Mandi, Guns, “Interior Point Solving for LP-based prediction+optimisation,” NeurIPS, 2020

SPO+: a deeper look at the (deep) learning

Standard:

Algorithm 1: Stochastic gradient descent

Input : training data $\mathcal{D} = \{X, y\}_{i=1}^{n}$, learning rate γ

1 initialize θ (neural network weights)
2 **for** *epochs* **do**
3 **for** *batches* **do**
4 sample batch $(X, y) \sim \mathcal{D}$
5 $\hat{y} \leftarrow g(z, \theta)$ (forward pass: compute predictions)
6 Compute loss $L(y, \hat{y})$ and gradient $\frac{\partial L}{\partial \theta}$
7 Update $\theta = \theta - \gamma \frac{\partial L}{\partial \theta}$ through backpropagation (backward pass)
8 **end**
9 **end**

with SPO+:

Algorithm 2: Stochastic gradient descent with SPO+ subgradient

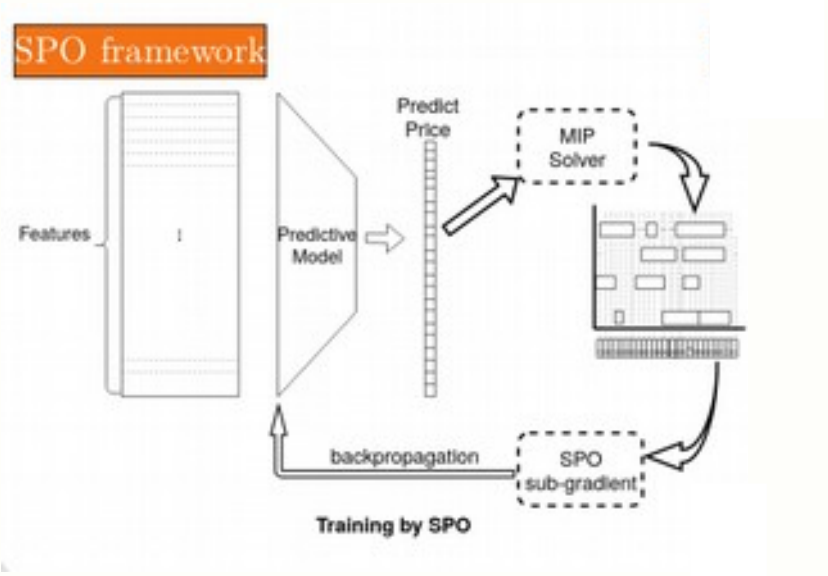
Input : training data $\mathcal{D} = \{X, y\}_{i=1}^{n}$, architecture g, learning rate γ

1 initialize θ (neural network weights of g)
2 **for** *epochs* **do**
3 **for** *batches* **do**
4 sample batch $(X, y) \sim \mathcal{D}$
5 $\hat{y} \leftarrow g(X, \theta)$ (forward pass: compute predictions)
6 $\bar{y} = y + 2(\hat{y} - y)$ // SPO+ trick, convex comb. of y and $\hat{y}$
7 Solve $sol = solver(\bar{y})$ // calls external solver
8 Use subgradient $\partial L = solver(y) - sol$
9 Update $\theta = \theta - \gamma \frac{\partial L}{\partial \theta}$ through backpropagation (backward pass)
10 **end**
11 **end**

we need to solve a CP on line 7 *for every training example*

(typically: 10-50 epochs, of 500 to 5000 samples...)

Can we do the solving better?

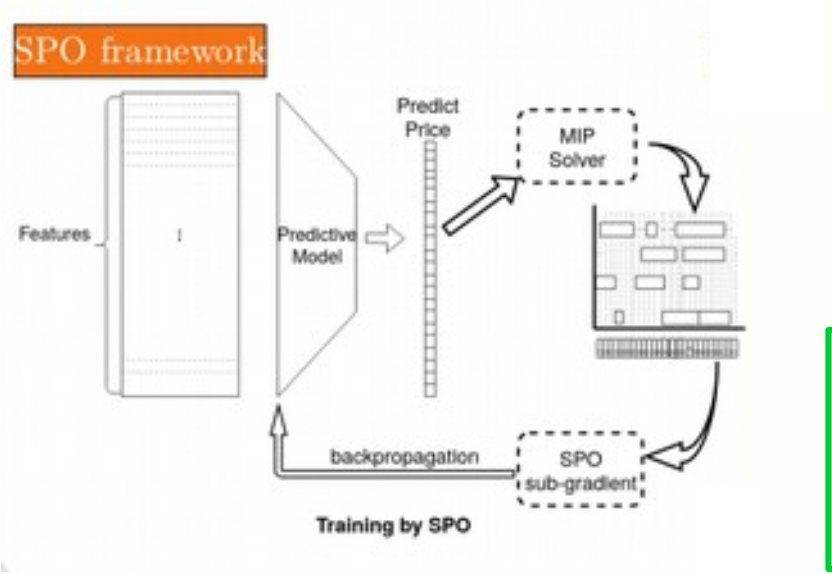


Challenge

To compute subgradient $v^*(2\hat{\theta} - \theta)$ must be solved repeatedly for each training instance

High training time & computation-expensive

Can we do the solving better?



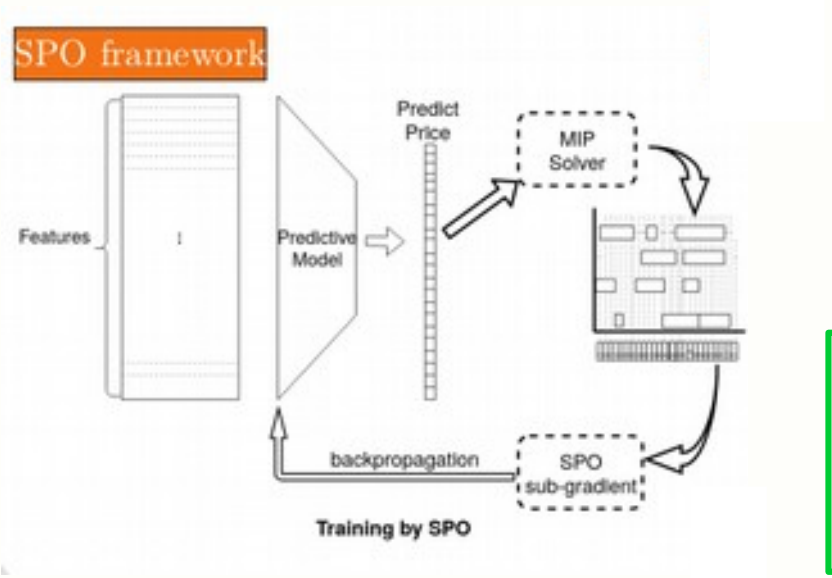
Challenge

To compute subgradient $v^*(2\hat{\theta} - \theta)$ must be solved repeatedly for each training instance

High training time & computation-expensive

Observe: constraints always the same,
only cost vector *y^* changes,
and we solve it for *thousands* of *y^* values,
each instance having a different true optimal solution

Can we do the solving better?



Challenge

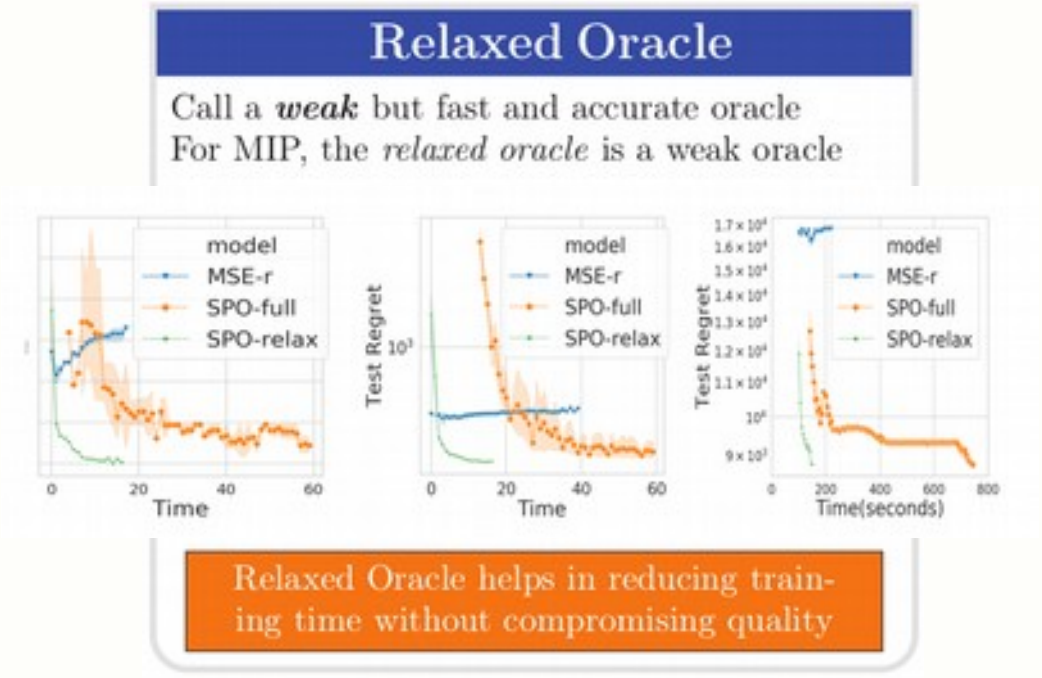
To compute subgradient $v^*(2\hat{\theta} - \theta)$ must be solved repeatedly for each training instance

High training time & computation-expensive

Observe: constraints always the same,
only cost vector *c* changes,
and we solve it for *thousands* of *c* values,
each instance having a different true optimal solution

- Solving MIP = repeatedly solving LP
  - Do we need to solve the MIP to optimality? or to a small gap?
  - Can we replace the MIP by the LP relaxation?
- Solving LP = repeatedly finding improved basis
  - Can we warm-start from previous basis's?

[Smart Predict-and-Optimize for Hard Combinatorial Optimization Problems, Jayanta Mandi, Emir Demirovic, Peter Stuckey, Tias Guns. AAAI20]



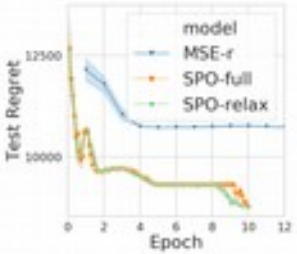
LP relaxations and warmstarts:

- Faster training time = possible to do wider grid search
- Faster training time = possible to scale up to larger problems

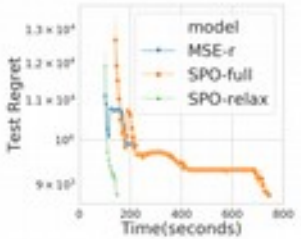
[Smart Predict-and-Optimize for Hard Combinatorial Optimization Problems, Jayanta Mandi, Emir Demirovic, Peter Stuckey, Tias Guns. AAAI20]

Relaxed Oracle

Call a ***weak*** but fast and accurate oracle
For MIP, the *relaxed oracle* is a weak oracle



(a) Epoch



(b) Time

Relaxed Oracle helps in reducing training time without compromising quality

SPO-relax is scalable

- Really hard instances:
  (1+ hour for single MIP solution)
- SPO-relax with total time budget:

Hard Instances (200 tasks on 10 machines)	Two-stage Approach				SPO-relax		
	2 epochs	4 epochs	6 epochs	8 epochs	2 hour	4 hour	6 hour
instance I	90,769	88,952	**86,059**	86,464	**72,662**	74,572	79,990
instance II	128,067	124,450	124,280	**123,738**	120,800	**110,944**	114,800
instance III	129,761	128,400	122,956	**119,000**	108,748	**102,203**	112,970
instance IV	135,398	132,366	132,167	**126,755**	109,694	99,657	**97,351**
instance V	122,310	**120,949**	122,116	123,443	118,946	**116,960**	118,460

[Smart Predict-and-Optimize for Hard Combinatorial Optimization Problems, Jayanta Mandi, Emir Demirovic, Peter Stuckey, Tias Guns. AAAI20]

Prediction + Optimisation for MIP

SPO's subgradient is an indirect 'black box' method

→ If we know it is a MIP… can we get better gradients?

Can we compute the gradient of a MIP?

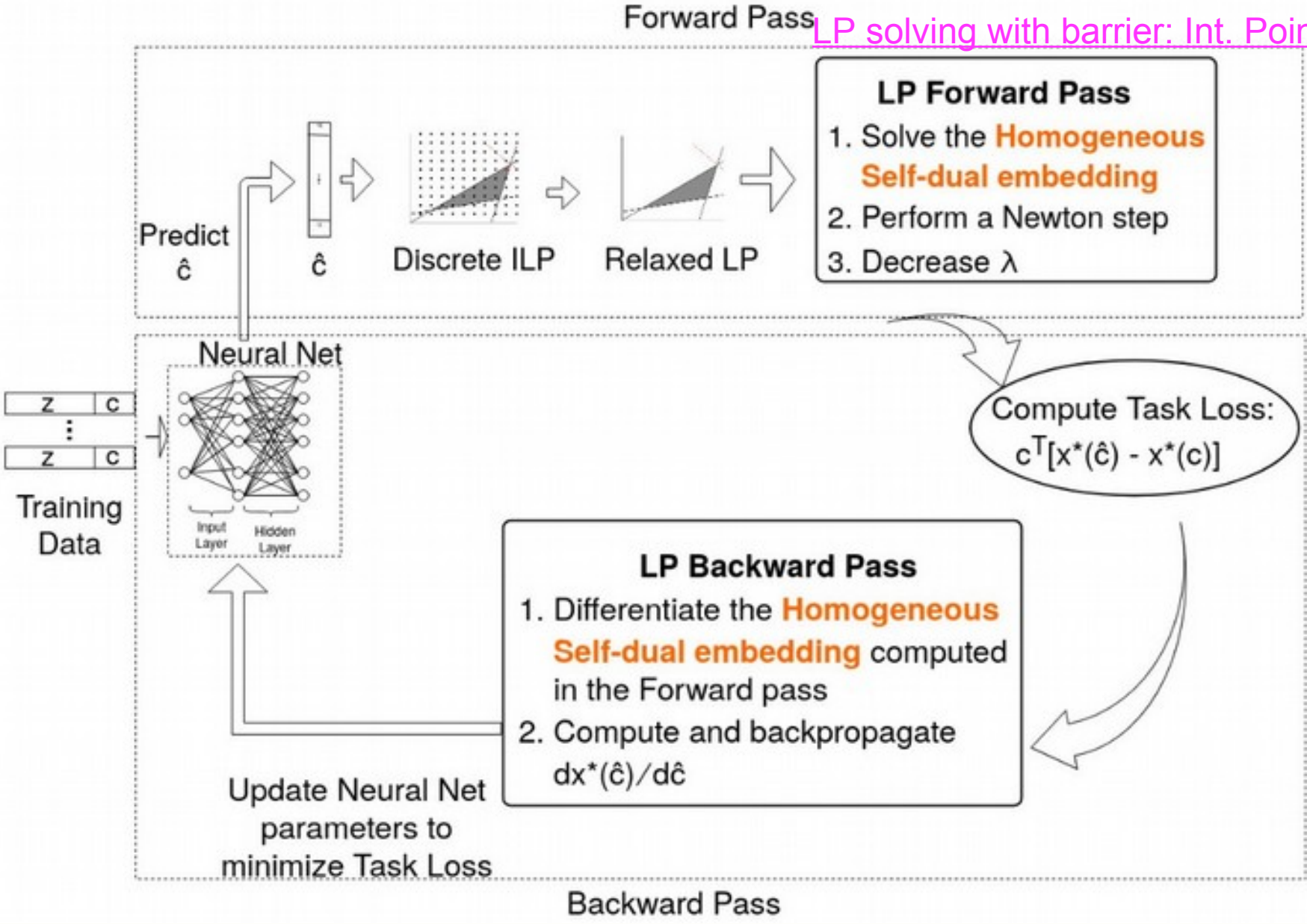
- Discrete so non-differentiable

Can we compute the gradient of an LP?

- Linear objective, so not twice differentiable
  - → Previous work: add quadratic term, differentiate QP
  - → Our work: add log barrier, differentiate *integer point* formulation!

["Interior Point Solving for LP-based prediction + optimisation", Jayanta Mandi, Tias Guns. NeurIPS20]

LP solving with barrier: Int. Point method



["Interior Point Solving for LP-based prediction + optimisation", Jayanta Mandi, Tias Guns. NeurIPS20]

Interior Point Solving for LP-based prediction + optimisation

KKT vs HSD

	KKT, log barrier			HSD, log barrier		
λ / λ-cut-off	10^{-1}	10^{-3}	10^{-10}	10^{-1}	10^{-3}	10^{-10}
Regret	*14365*	14958	21258	**10774**	14620	21594

Table 1: Differentiating the HSD formulation is more efficient than differentiating the KKT condition

Compariosn with the state of the art

	Two-stage		QPTL		SPO		HSD,log barrier	
	0-layer	1-layer	0-layer	1-layer	0-layer	1-layer	0-layer	1-layer
MSE-loss	**745** (**7**)	796 (5)	3516 (56)	2×10^9 (4×10^7)	3327 (485)	3955 (300)	2975 (620)	1.6×10^7 (1×10^7)
Regret	13322 (1458)	13590 (2021)	13652 (325)	13590 (288)	11073 (895)	12342 (1335)	**10774** (**1715**)	11406 (1238)

Table 2: Our approach is able to outperform the state of the art

["Interior Point Solving for LP-based prediction + optimisation", Jayanta Mandi, Tias Guns. NeurIPS20]

Prediction + Optimisation for MIP and more

All current method use a 'continuous approximation' to make it non-discrete and hence (almost) differentiable

Observation: constraints always stay the same, so the polytope is always the same.

→ Can we also use an inner approximation?

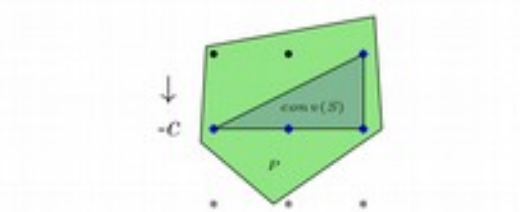


Figure 1: Representation of the solution pool and the continuous relaxation of set V.

[“Discrete solution pools and noise-contrastive estimation for predict-and-optimize” Maxime Mulamba, Jayanta Mandi, Michelangelo Diligenti, Michele Lombardi, Victor Bucarey, Tias Guns, arxiv 2020]

Prediction + Optimisation for MIP and more

All current method use a 'continuous approximation' to make it non-discrete and hence (almost) differentiable

Observation: constraints always stay the same, so the polytope is always the same.

→ Can we also use an <u>inner approximation</u>?

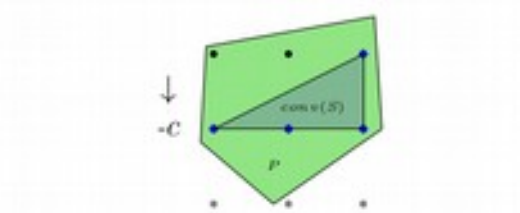


Figure 1: Representation of the solution pool and the continuous relaxation of set V.

<u>Inner approximation = pool of known solutions</u>

→ can replace 'solver()' by 'argmax()' over finite solutions! (SPO+)

→ can use probabilistic loss function (Noise-Contrastive Estimation)

[“Discrete solution pools and noise-contrastive estimation for predict-and-optimize” Maxime Mulamba, Jayanta Mandi, Michelangelo Diligenti, Michele Lombardi, Victor Bucarey, Tias Guns, arxiv 2020]

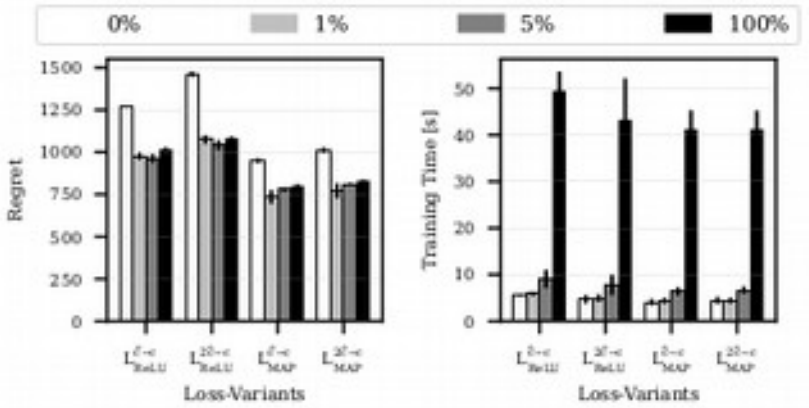
Prediction + Optimisation for MIP and more

Inner approximation = pool of known solutions

→ can replace 'solver()' by 'argmax()' over finite solutions! (SPO+)

→ can use probabilistic loss function (Noise-Contrastive Estimation)

Main advantage: do not have to call a solver for each training instance! Can 'grow' solution pool **FAST and GOOD**



["Discrete solution pools and noise-contrastive estimation for predict-and-optimize" Maxime Mulamba, Jayanta Mandi, Michelangelo Diligenti, Michele Lombardi, Victor Bucarey, Tias Guns, arxiv 2020]

It's so fast we can actually train a convolutional neural network THROUGH a CP problem...

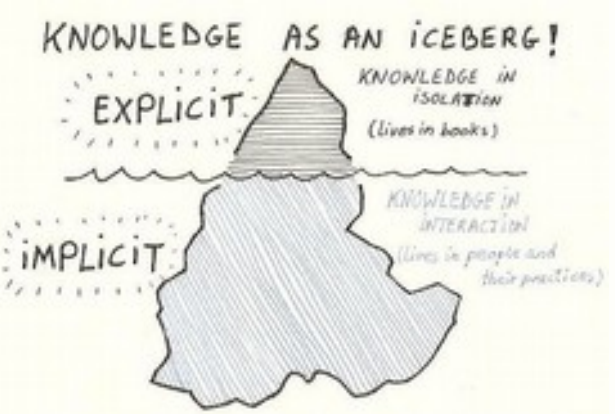
Visual sudoku :)

	grid-acc	givens-acc	cell-acc
Two-stage	87.50%	98.00%	98.09%
Blackbox-Pool (10%)	54.00%	89.62%	88.13%
$\mathcal{L}_{MAP}^{(\hat{c}-c)}$ (10%)	74.50%	94.37%	94.51%
SPO-Pool (10%)	66.75%	92.67%	92.88%

Table 5: Classification task accuracy

For this task, training individual digits still better.

Key take-aways:



- <u>Explicit</u> knowledge: use solver
- <u>Implicit</u> knowledge: do learning

- Joint inference / collective classification:
  maximize log likelihood!
- Keep revisiting the solving AND the learning,
  hybridize and use properties of one in the other!
- Comb. optimisation inside neural loss becoming actually feasible
  → end-to-end hybrid prediction and optimisation

Future Work

- Complexity of learned models vs. complexity of CP solving
- Faster (runtime), more accurate learning
- Interactive preference learning, multi-agent
- Other perception data (language, voice, camera)

- Wide range of applications (Industry 4.0, transport & more)