On the combination of knowledge and learning

Andrea Passerini

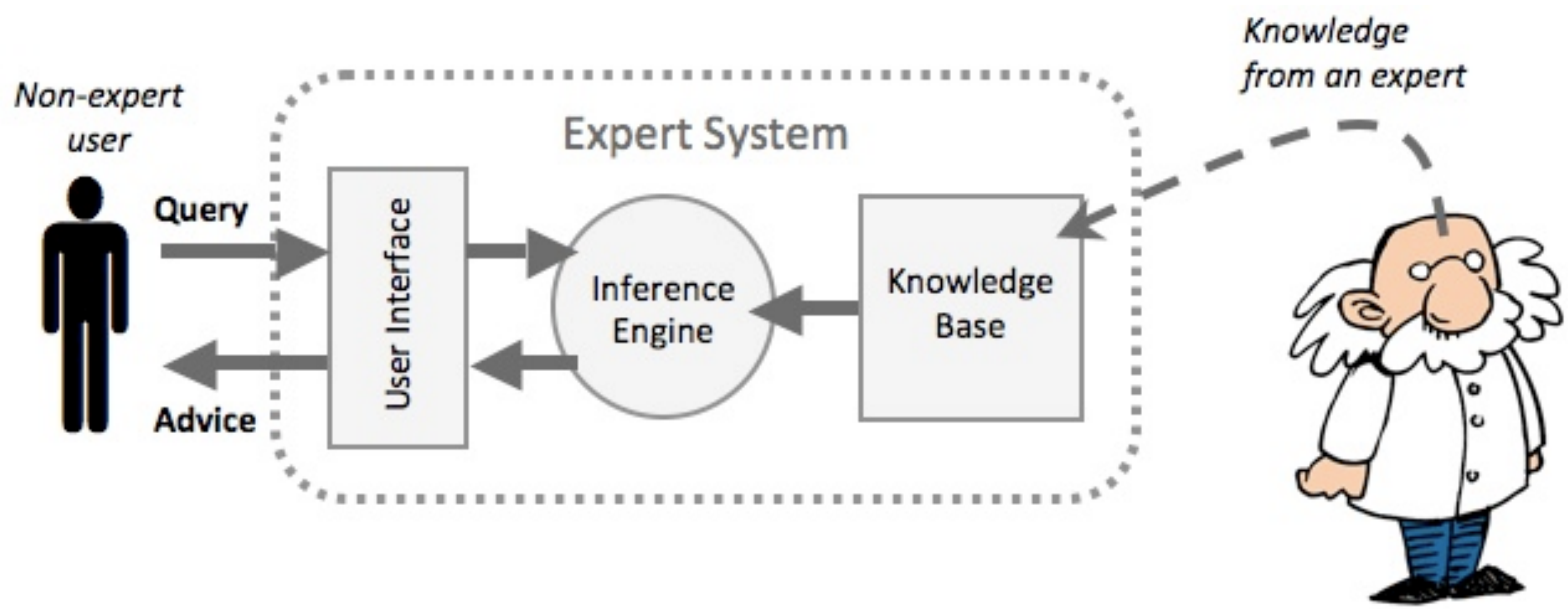
UNIVERSITY
OF TRENTO - Italy

KSEM- 2019

The first AI boom: **expert systems**

symbolic reasoning

knowledge acquisition



knowledge base

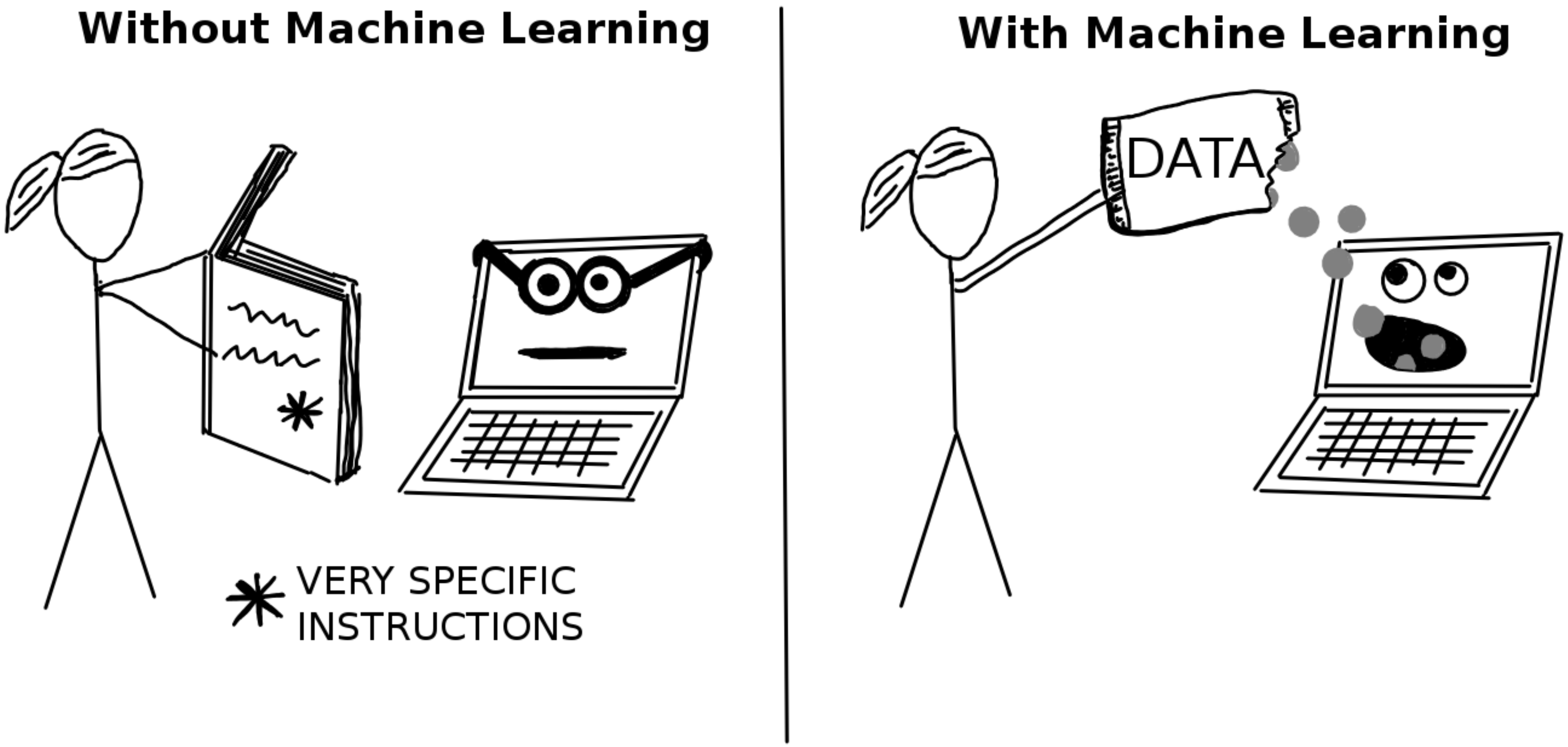
Limitations of expert systems: **AI winter**

knowledge acquisition bottleneck

data integrity

no management of uncertainty

New AI spring: **machine learning**

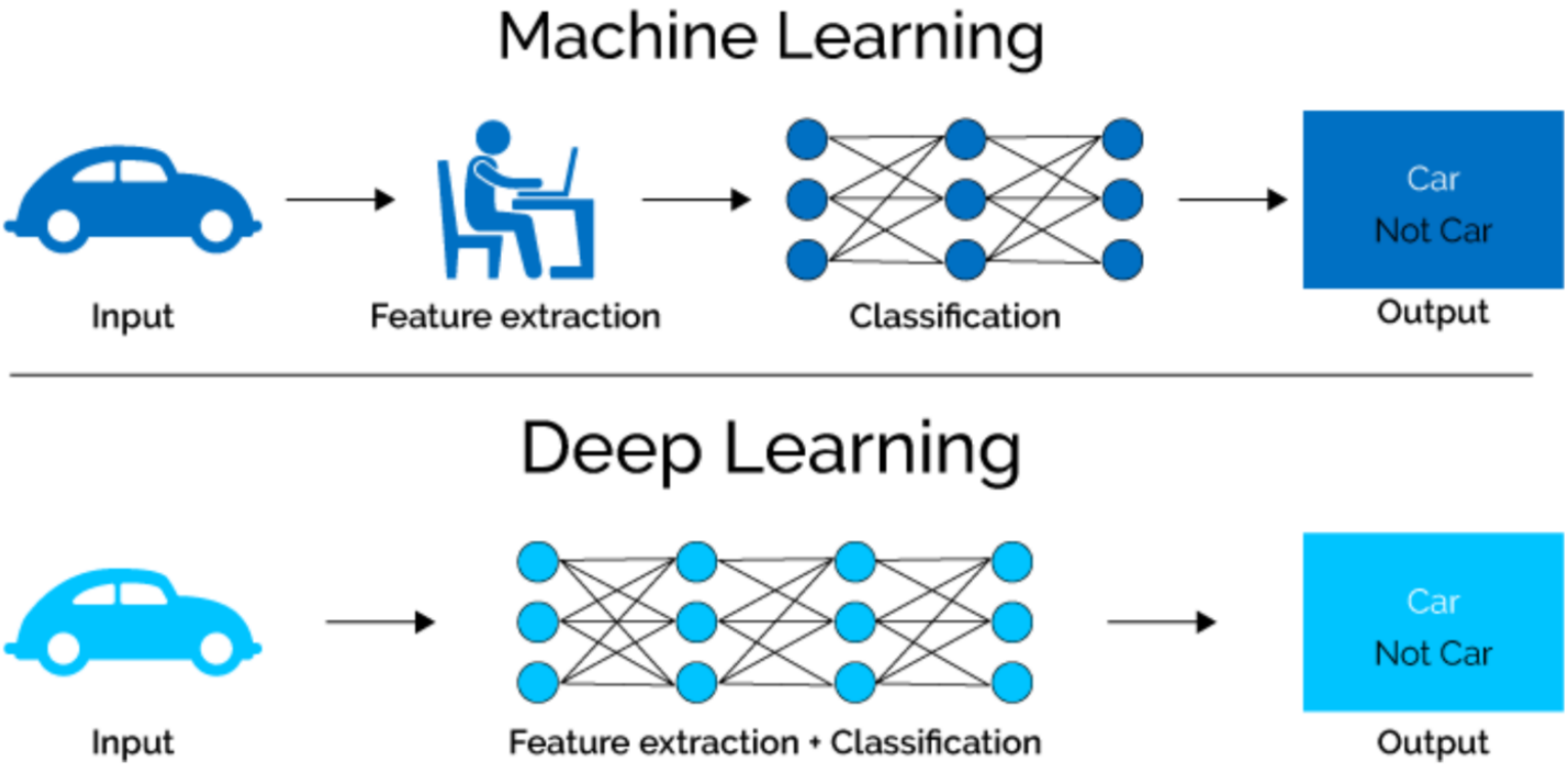


uncertainty management

noise robustness

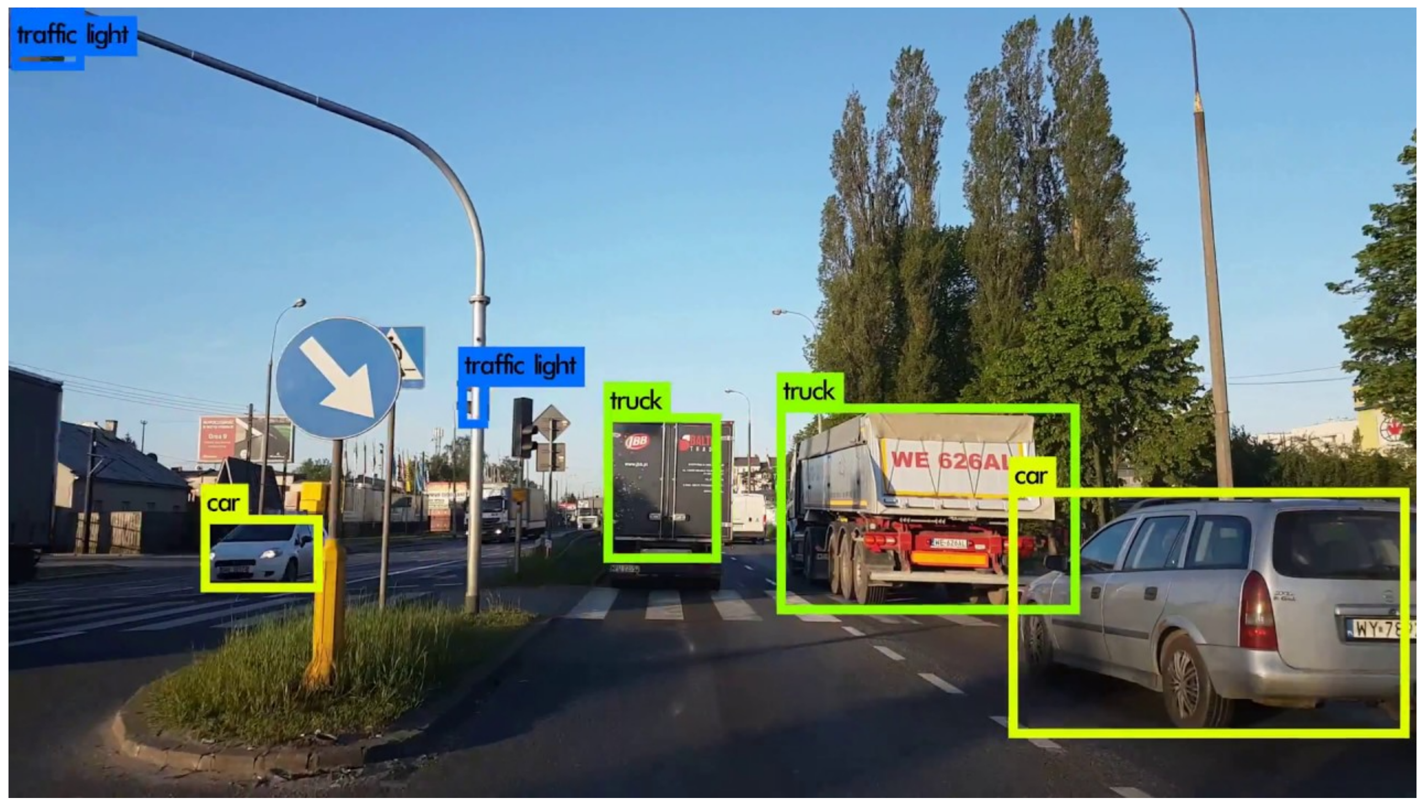
learning from data

Current AI boom: **deep learning**



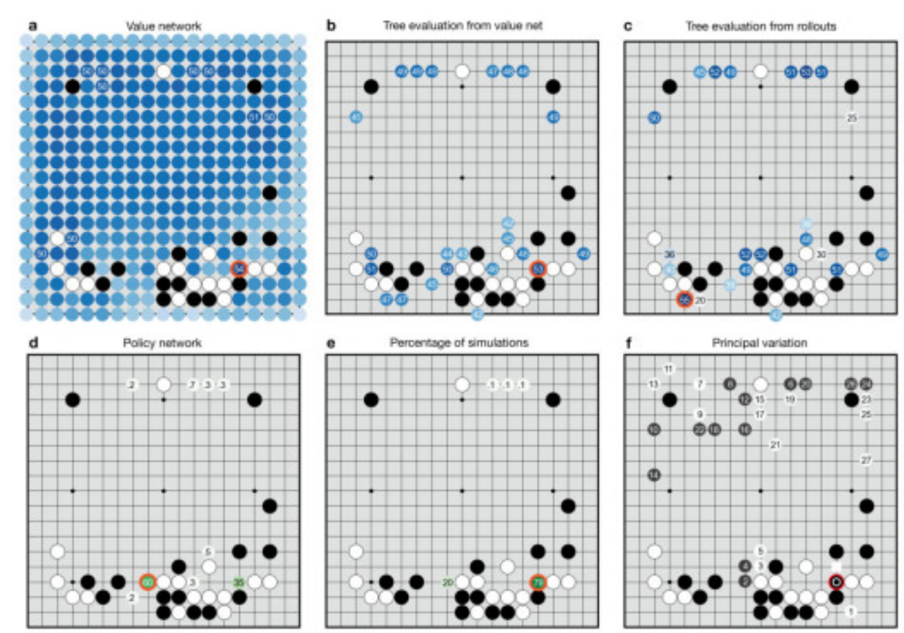
Deep learning successes

computer vision



game playing

image generation



AlphaGo

Limitations of deep learning

- **Data hungry**: need tons of examples to learn
- **Opaqueness**: learn a black box model
- **Lack of Verifiability**: no formal guarantees on performance
- **Lack of Flexibility**: problems in adapting to novel information

(in)Famous DL failures

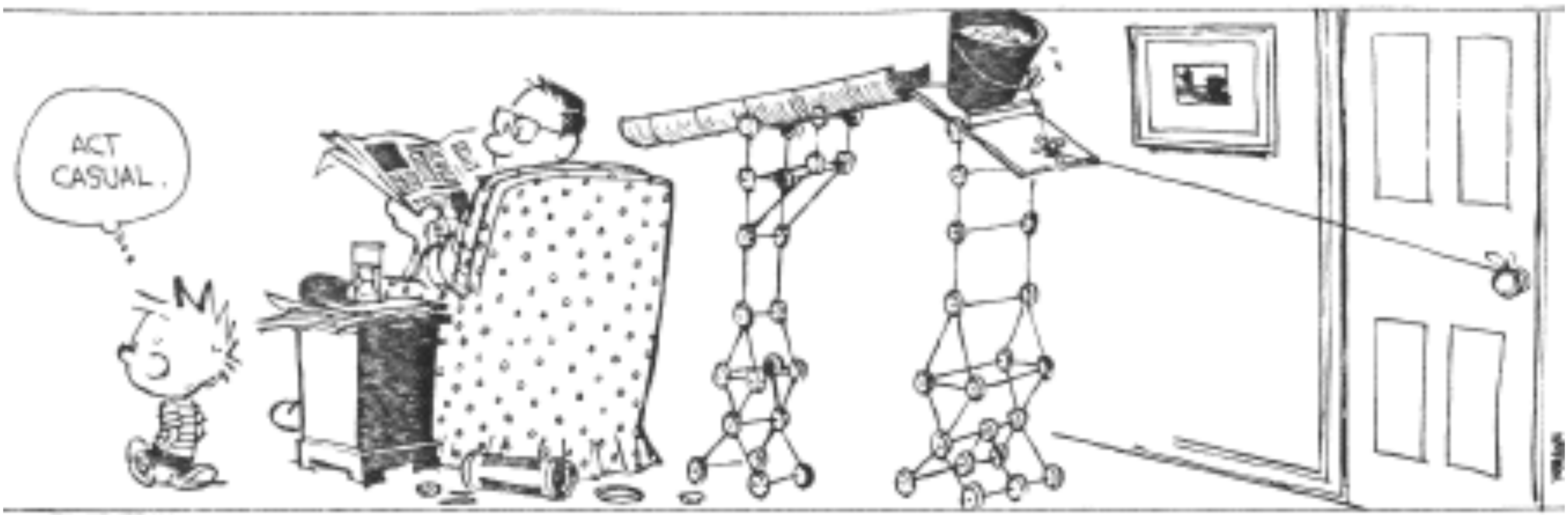
- Microsoft’s AI **chatbot corrupted by Twitter Trolls** into a pro-nazi mouthpiece
- **Photo of Chinese billionaire on an** ad on a passing bus wrongly **identified as a jaywalker**
- Google photos labels **two black people as ‘gorillas’**
- Amazon Alexa places an **order for 170$ dollhouse, when a six-years old asked** for one

Solution: **learning + knowledge**

Explainable AI

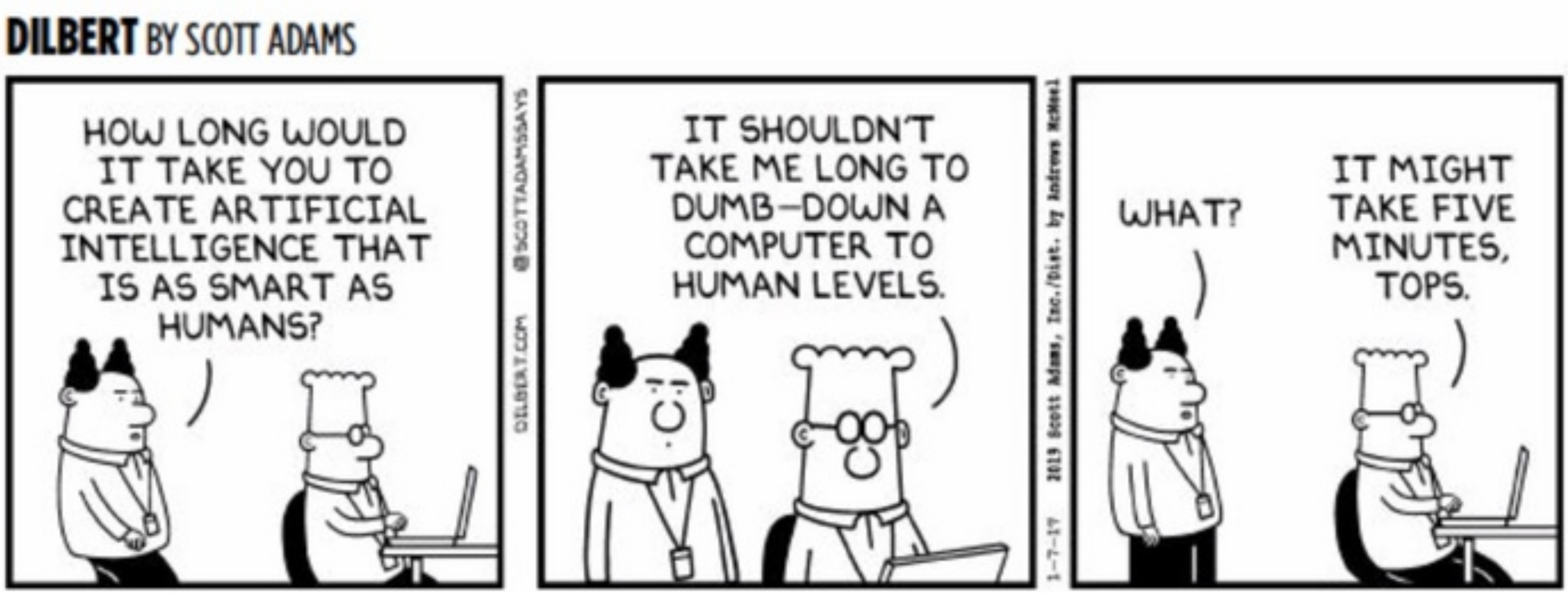
Solution: **learning + knowledge**

verifiable AI



Solution: **learning + knowledge**

Human-level AI



Plenty of approaches exist

Bayesian logic
(Deep) Problog
Probabilistic logic learning
Stochastic logic programs
Prism
Markov logic networks

Probabilistic Soft Logic
Logic tensor networks
Statistical learning + fuzzy logic
Semantic-based regularisation

Neural Turing machine
Neural reasoning
Neural theorem proving
Differentiable neural computer

ProbLog

(De Raedt & Kimmig, 2015)

```
0.2::red.

flip(coin1).
flip(coin2).

0.3::heads(coin1).
0.2::heads(coin2).

side(X,heads) :- heads(X).

is_heads :- flip(X), side(X,heads).

win :- is_heads.
win :- \+is_heads, red.

query(win).
```

DeepProbLog

(Manhaeve et al, 2018)

neural network taking coin image as input and returning probability distribution over flip outcomes

```
0.2::red.

flip(coin1).
flip(coin2).

%0.3::heads(coin1).
%0.2::heads(coin2).

nn(pred_side, X, [heads, tails])::side(X,heads);side(X,tails).

is_heads :- flip(X), side(X,heads).

win :- is_heads.
win :- \+is_heads, red.

query(win).
```

Semantic based regularisation (SBR)

(Diligenti, Gori, Saccà, 2017)

regularisation term

number of **tasks**

training error loss

$$\lambda_r \boxed{\sum_{k=1}^{T} ||f_k||^2} + \boxed{\sum_{k=1}^{T} \sum_{\left(\boldsymbol{x}_k^j, y_k^j\right) \in \mathcal{L}_k} L\left(f_k(\boldsymbol{x}_k^j), y_k^j\right)} + \lambda_c \boxed{\sum_{h=1}^{H} \phi_h(\mathcal{S}, \boldsymbol{f})}$$

predictor for task *k*

constraint satisfaction term

Deep SBR (Lyrics)

(Marra et al., 2019)

regularisation term

number of **tasks**

training error loss

$$\lambda_r \boxed{\sum_{k=1}^{T} ||f_k||^2} + \boxed{\sum_{k=1}^{T} \sum_{\left(\boldsymbol{x}_k^j, y_k^j\right) \in \mathcal{L}_k} L\left(f_k(\boldsymbol{x}_k^j), y_k^j\right)} + \lambda_c \boxed{\sum_{h=1}^{H} \phi_h(\mathcal{S}, \boldsymbol{f})}$$

deep NN predictor for task *k*

constraint satisfaction term

My research: **learning with constraints**

- **Learning + constraint solving**
  - to deal with complex background knowledge
- **Interactive machine learning**
  - to keep the user in the loop of the learning process
- **Constructive machine learning**
  - i.e. learning to synthesise novel entities from scratch
- **Learning constraints**
  - to automatically extract knowledge from data

Prediction as constrained optimisation

given part defining **input/ context** (possibly empty)

unknown part defining **output/solution**

$$y^* = \underset{y \in \mathcal{Y}_{\text{feasible}}}{\operatorname{argmax}} \underbrace{\langle \boldsymbol{w}, \boldsymbol{\varphi}(x, y) \rangle}$$

set of constraints defining feasible space

scoring function defining solution quality

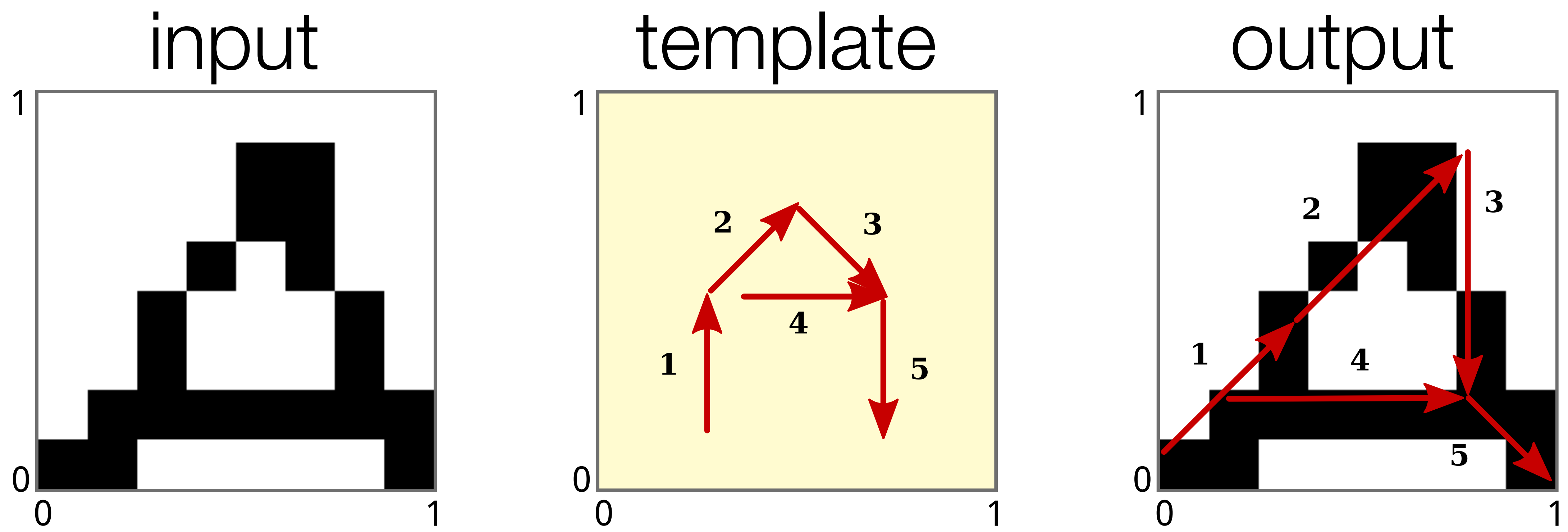
feature function defining relevant features

Learning paradigms

- **Structured-output learning**
  - oracle provides best output for given input
- **Preference elicitation**
  - oracle provides feedback on candidate output

Structured Learning Modules Theories

[Artificial Intelligence Journal, 2017]



- **Left**: 8x8 B/W bitmap image of an "A"
- **Middle**: vectorial representation of an "A"
- **Right**: vectorial representation fitted on the image

Problem formalisation

- **Input**: set of pixels belonging to character
- **Output**: set of ***m*** directed segments (represented by their begin and end coordinates)

- **Score**:

$$\operatorname*{argmax}_y \left(coverage(x, y), orientation(y) \right) \boldsymbol{w}$$

  - output should **cover** character pixels
  - output should *"**resemble**" corresponding vectorial template*

Coverage

- Coverage is fraction of covered pixels

$$coverage(x, y) := \frac{1}{|P|} \sum_{p \in P} \mathbb{1}(\texttt{covered}(p))$$

- Pixel covered by at least one segment

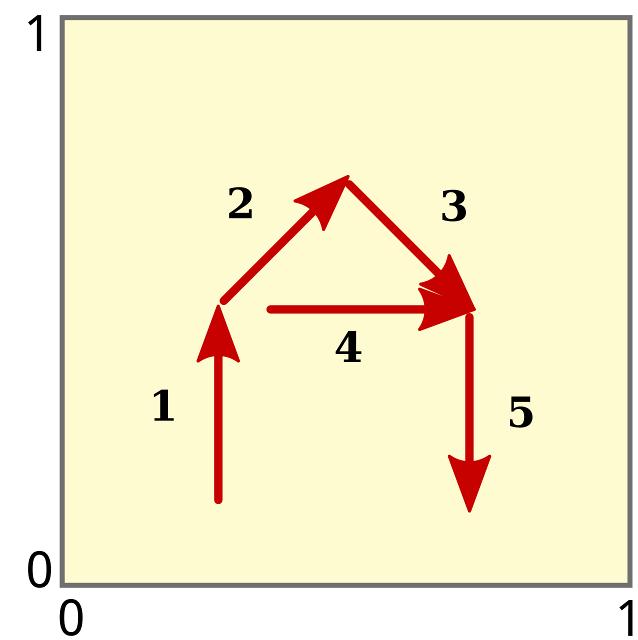
$$\texttt{covered}(p) := \bigvee_{i \in [1,m]} \texttt{covered}(p, i)$$

- Pixel coverage formula depends on segment orientation

Orientation

- Contains features about orientation of segments
- E.g. An “A” could have a vectorial representation like:

$$\texttt{increasing}(1) \wedge \texttt{head2tail}(1,2) \wedge \texttt{increasing}(2) \wedge \texttt{head2tail}(2,3) \wedge \texttt{decreasing}(3) \wedge \texttt{head2head}(3,4) \wedge \texttt{horizontal}(4) \wedge \texttt{head2tail}(4,5) \wedge \texttt{decreasing}(5)$$



- Character template is **not available** at test time
- Orientation features represent **all possible** orientations and connections for segments

Scoring function

- **Scoring function** weighted combination of coverage and orientation features

$$
\begin{aligned}
score := \boldsymbol{w}^\top (& \underbrace{increasing(i), decreasing(i), right(i)}_{\text{for all segments } i}, \\
& \underbrace{h2t(i,j), t2h(i,j), h2h(i,j), t2t(i,j)}_{\text{for all segments } i,j \text{ with } i<j}, \\
& coverage)
\end{aligned}
$$

- Appropriate **weights for the character** should be learned

Structured-output learning **problem**

margin term

penalty term

$$\min_{\mathbf{w},\boldsymbol{\xi}} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i$$

penalty for not satisfying constraint

subject to:

$$\boldsymbol{w}^\top(\boldsymbol{\psi}(x_i, y_i) - \boldsymbol{\psi}(x_i, y')) \geq \Delta(y_i, y') - \xi_i$$

$$\forall\, i = 1, \ldots, n;\; y' \neq y_i$$

correct solution better than any incorrect one

difference between correct and incorrect output

Problem: exponential number of constraints!!

Structured-output learning **problem**

margin term

penalty term

$$\min_{\mathbf{w},\boldsymbol{\xi}} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i$$

penalty for not satisfying constraint

subject to:

$$\boldsymbol{w}^\top(\boldsymbol{\psi}(x_i, y_i) - \boldsymbol{\psi}(x_i, y')) \geq \Delta(y_i, y') - \xi_i$$

$$\forall\, i = 1, \ldots, n;\; y' \neq y_i$$

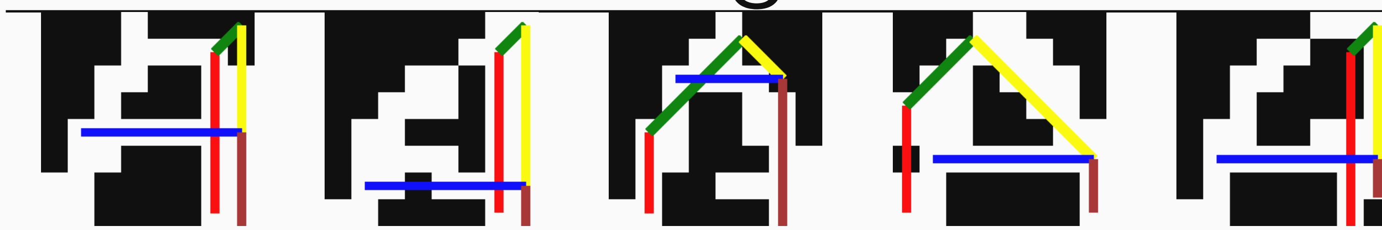
correct solution better than any incorrect one

difference between correct and incorrect output

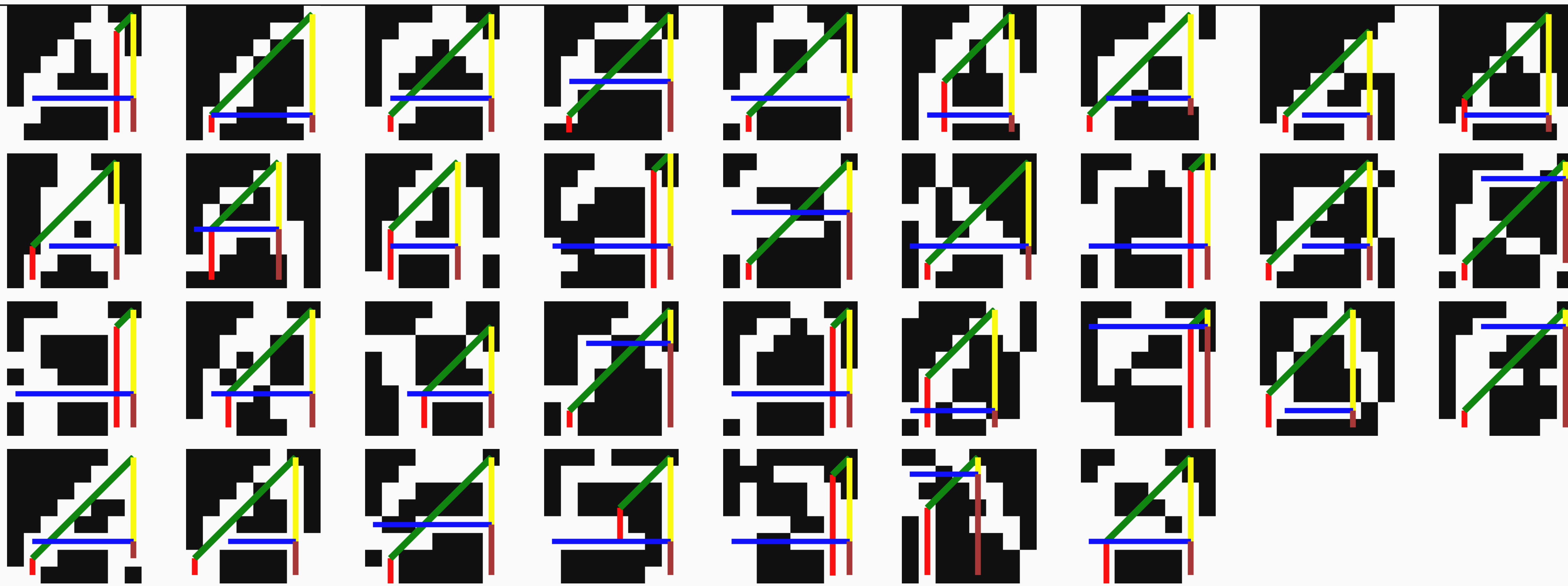
Solution: iterative approaches (cutting plane, Frank-Wolfe)

Results for “A”

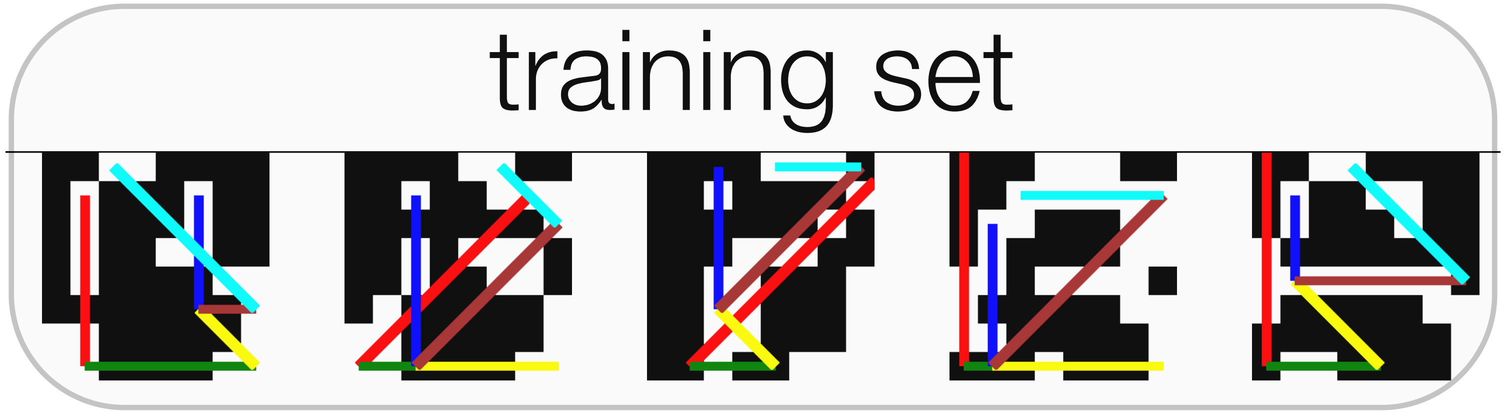
training set

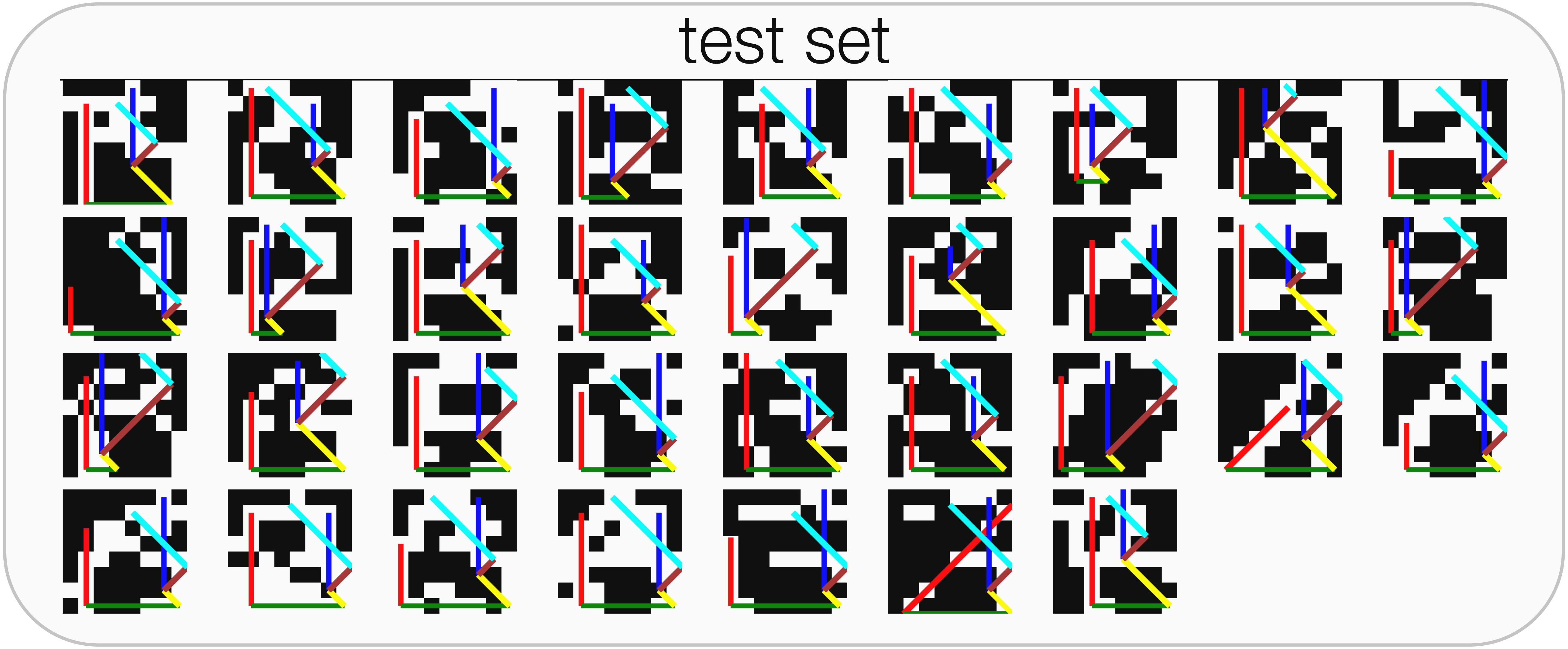


test set



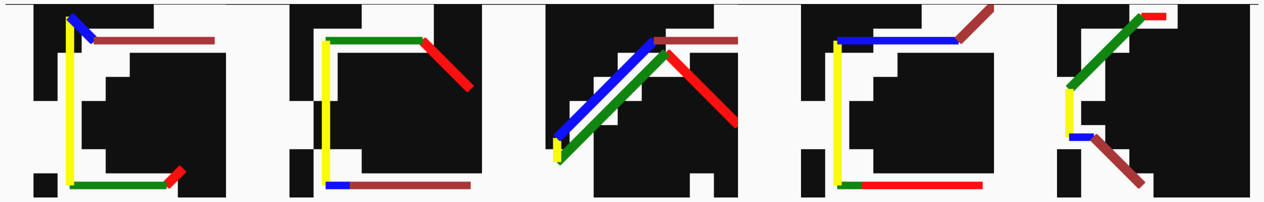
Results for “B”



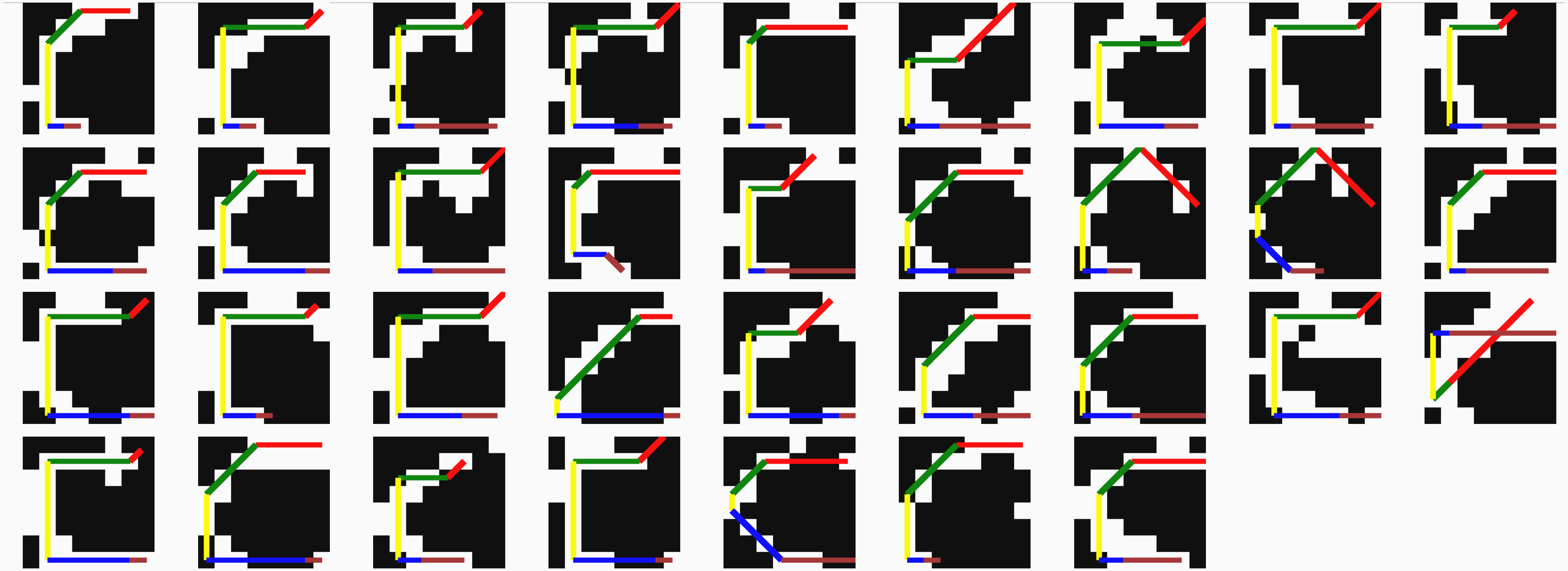


Results for “C”

training set

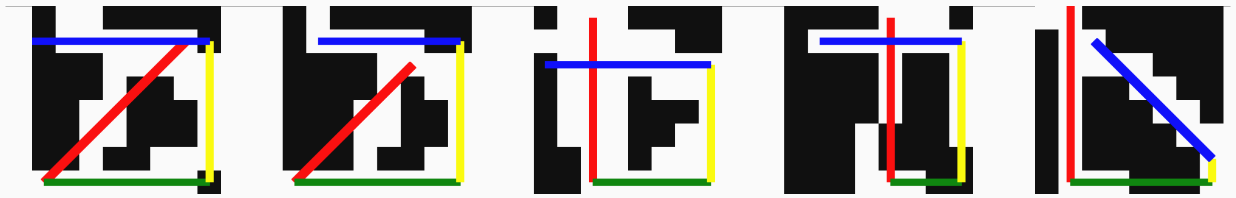


test set

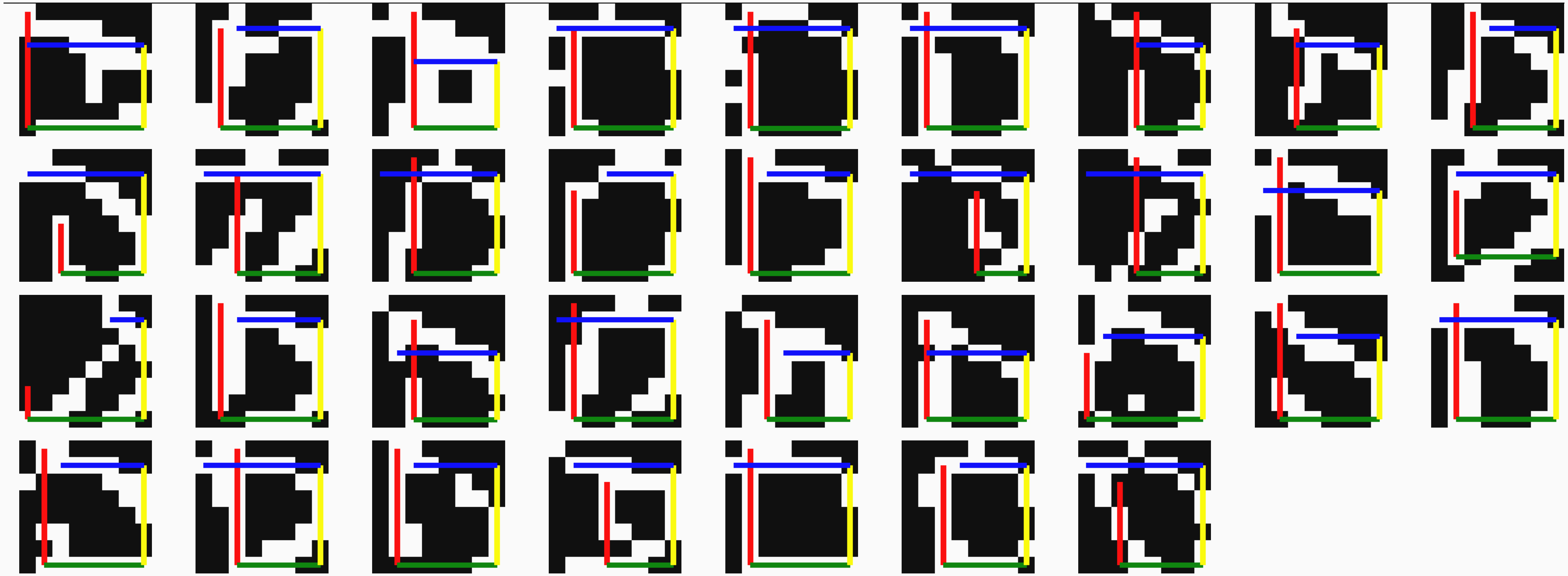


Results for “D”

training set

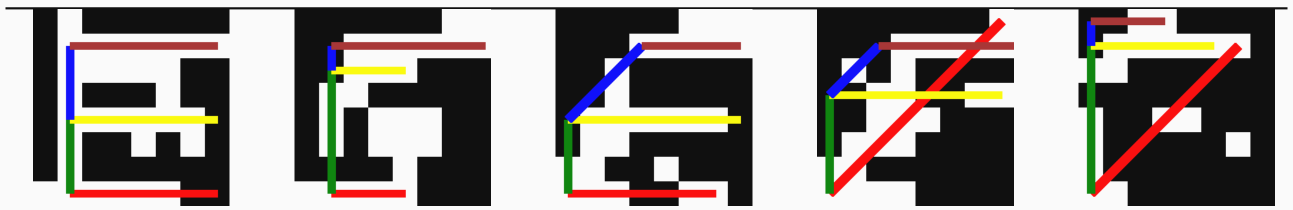


test set

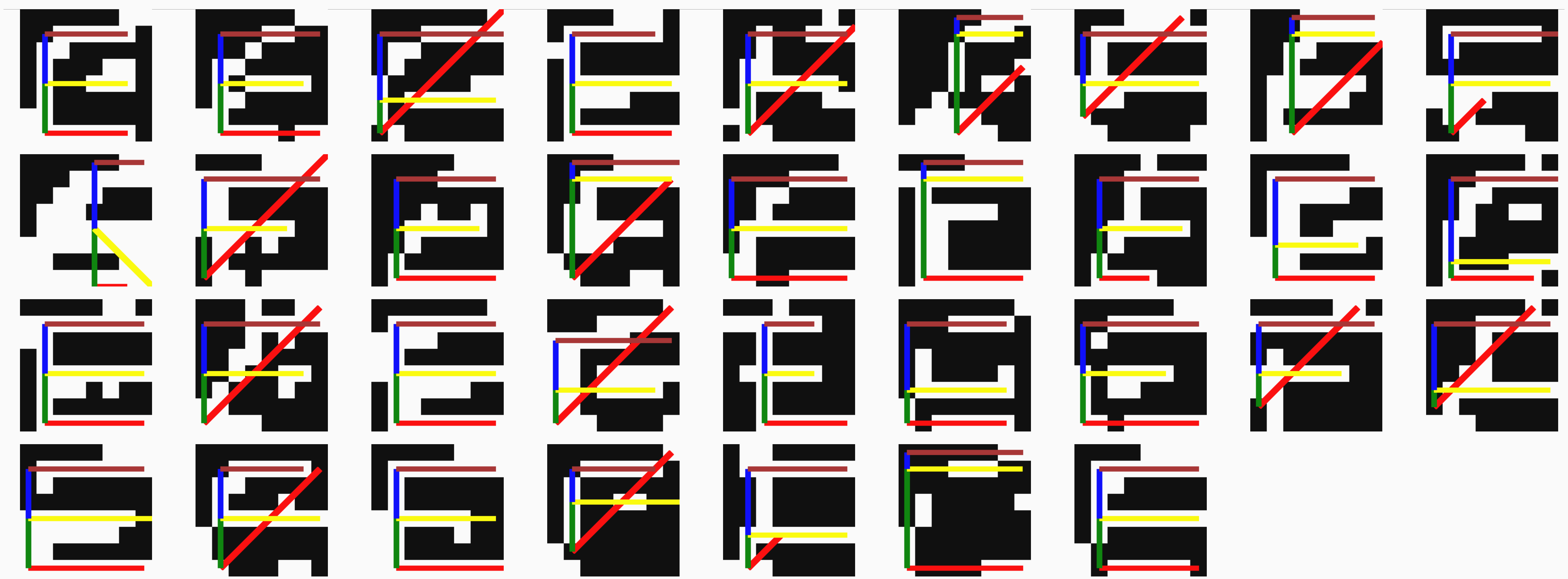


Results for “E”

training set



test set



Pyconstruct:
a library for declarative structured-output learning

[IJCAI 2018]

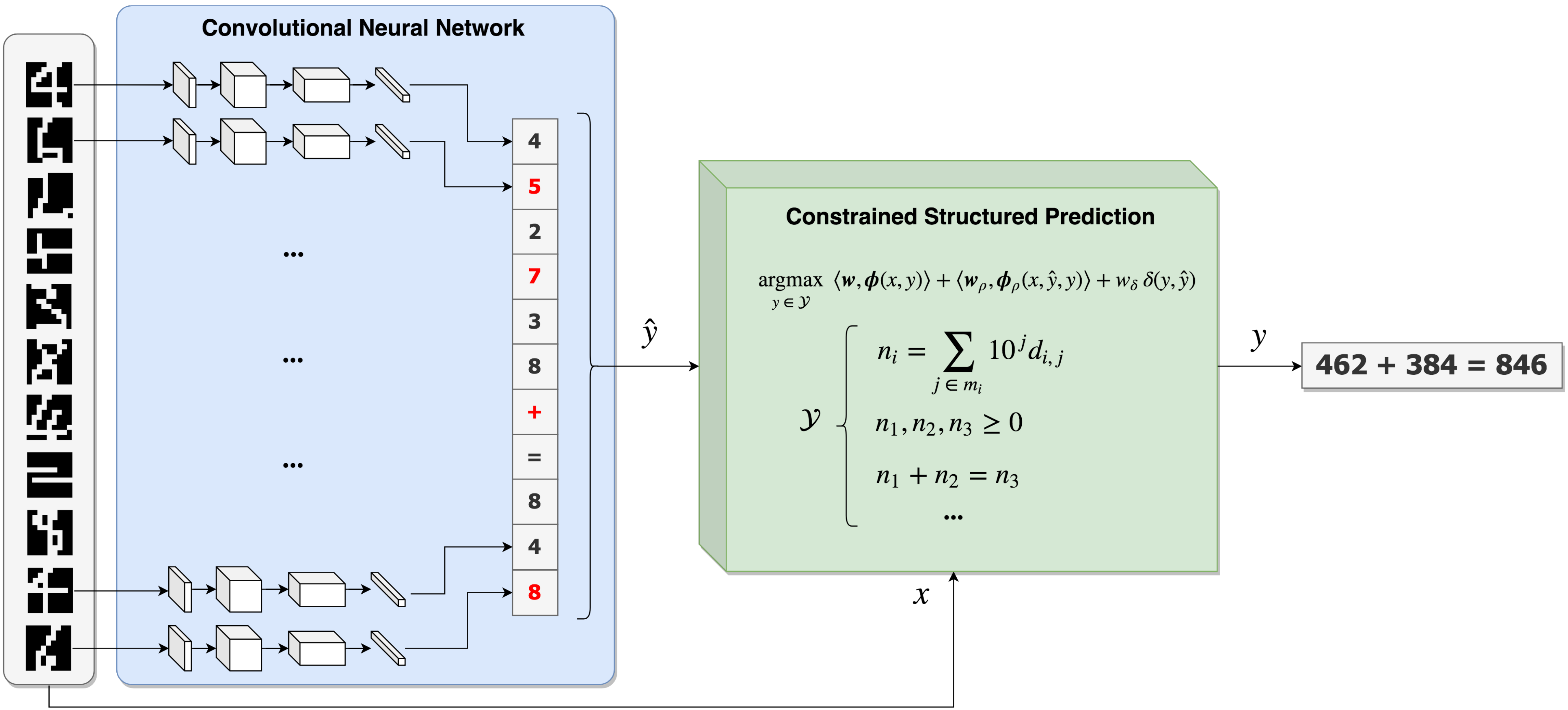
Pyconstruct

Pyconstruct is a Python library for declarative, constrained, structured-output prediction. When using Pyconstruct, the problem specification can be encoded in MiniZinc, a high-level constraint programming language. This means that domain knowledge can be declaratively included in the inference procedure as constraints over the optimization variables.

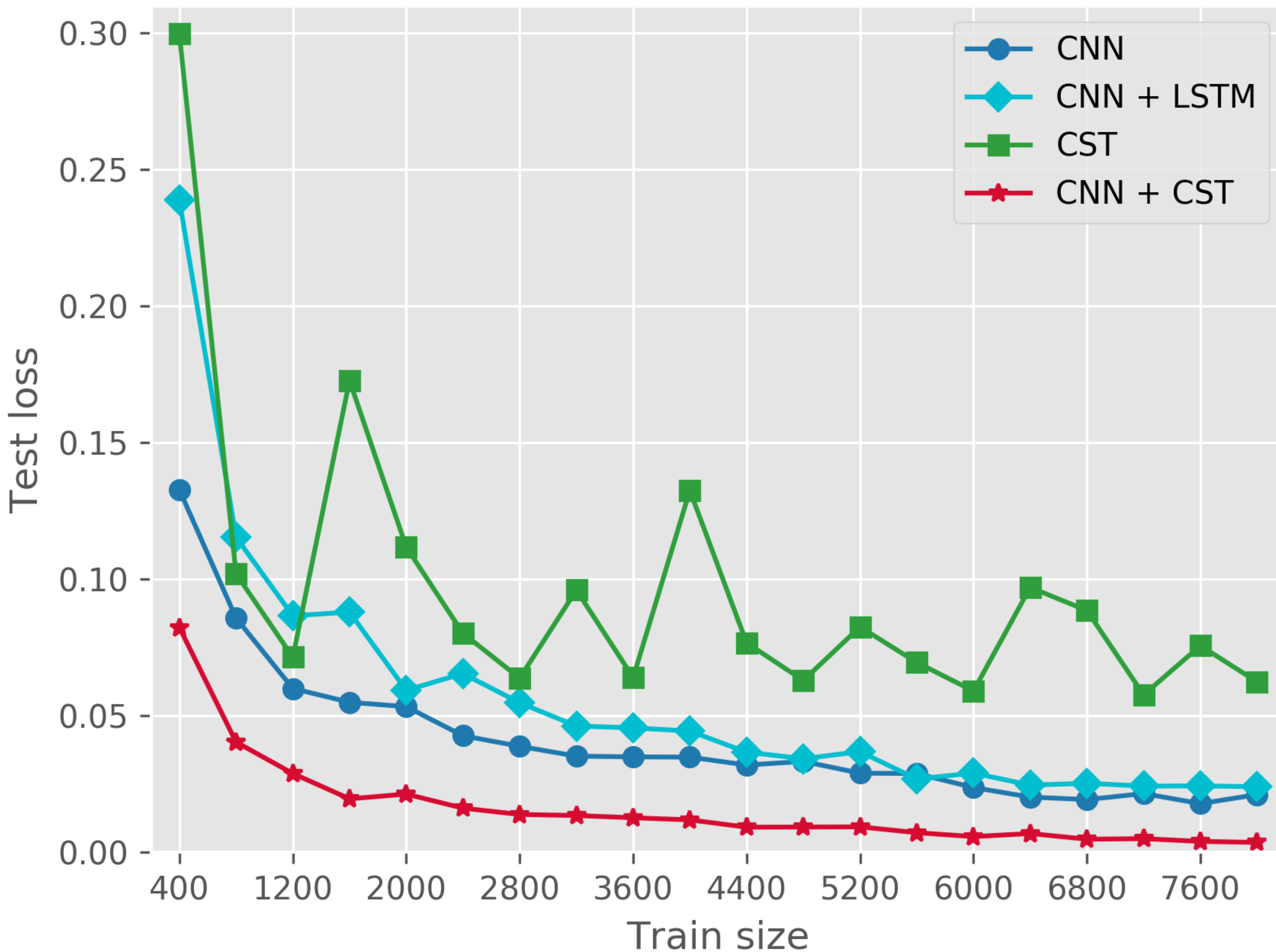
Check out the Quick Start guide to learn how to solve your first problem with Pyconstruct.

Have a look at the docs and the reference manual too, to learn more about it!

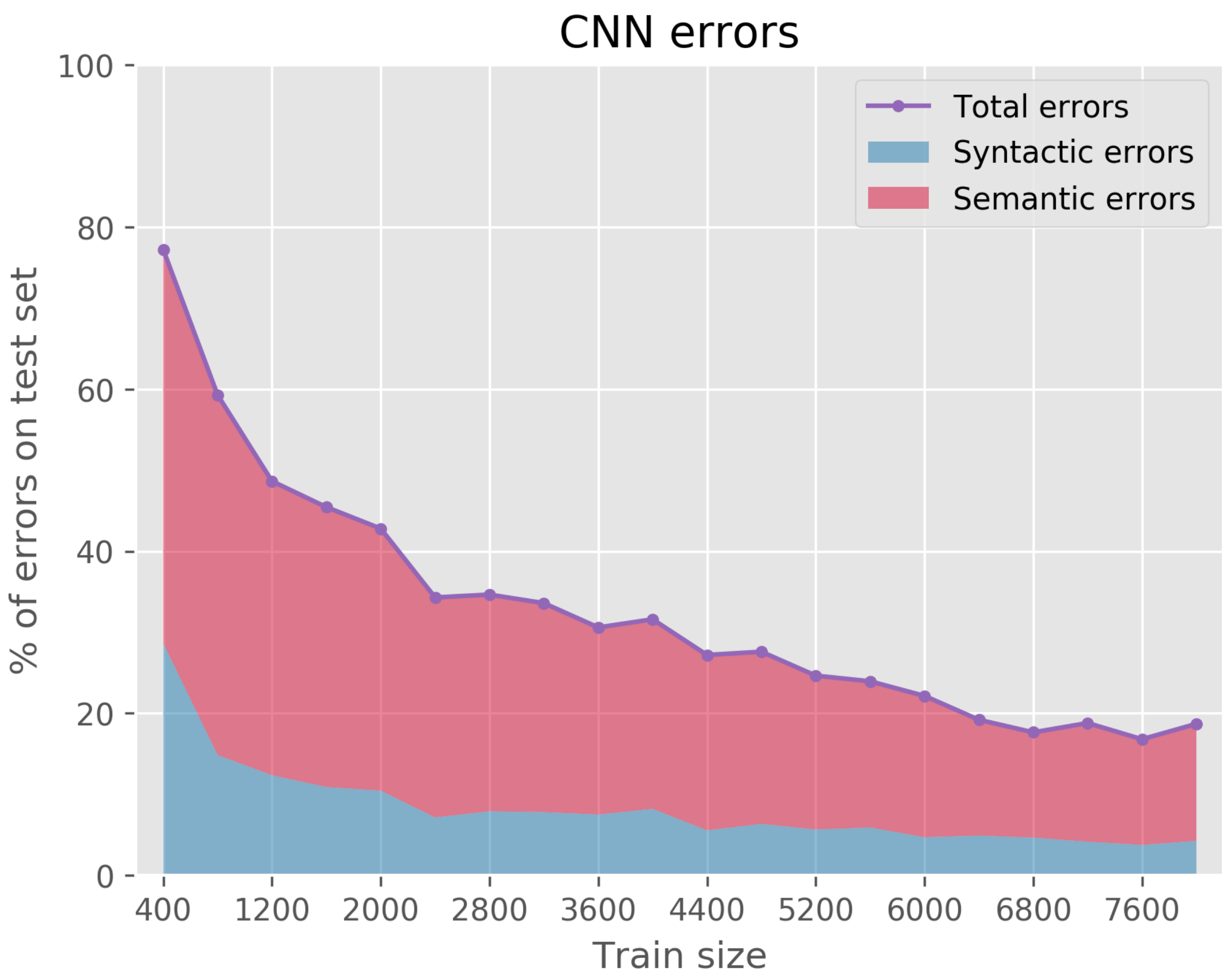
Going deep:
Deep declarative structured-output learning

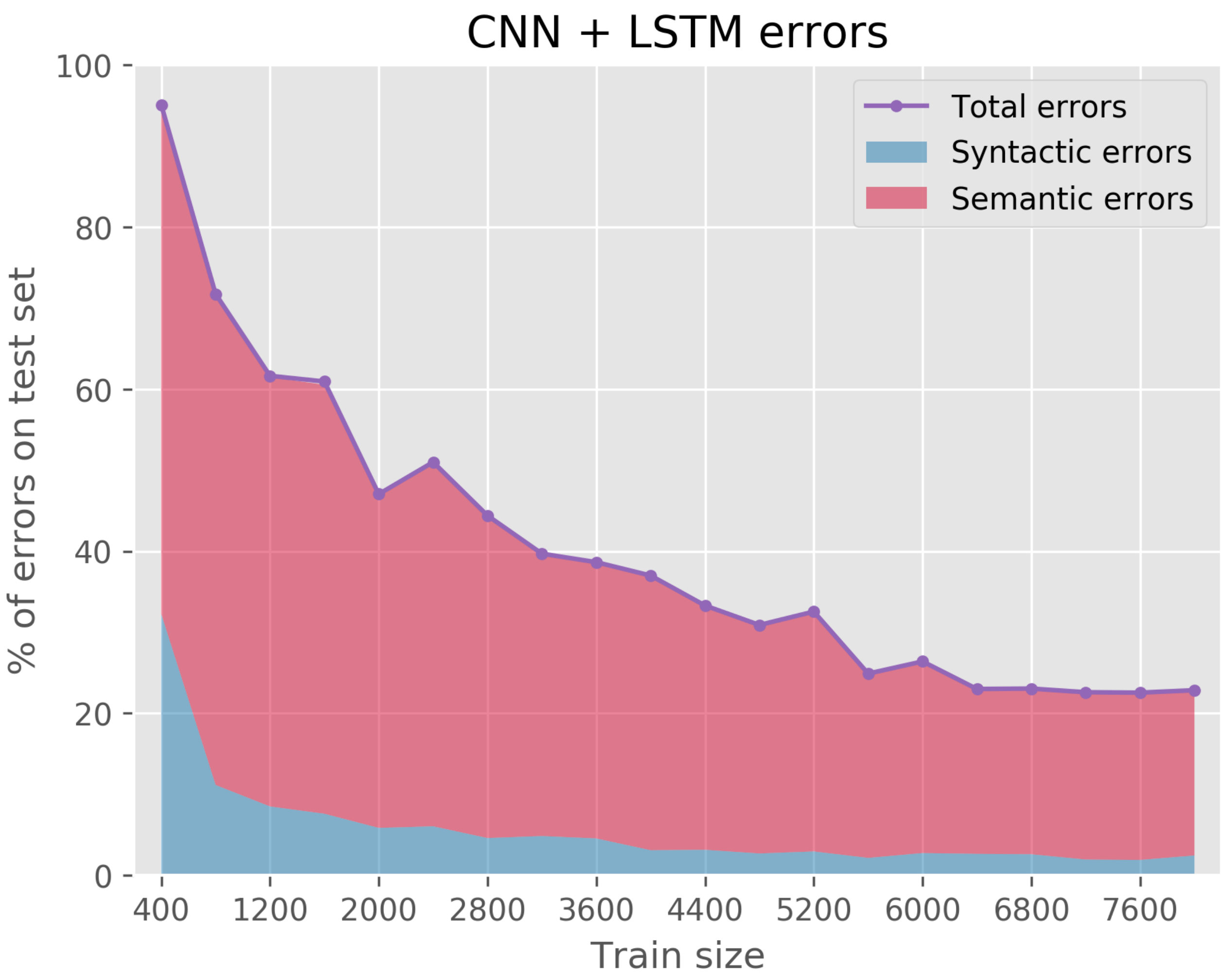


Experimental results



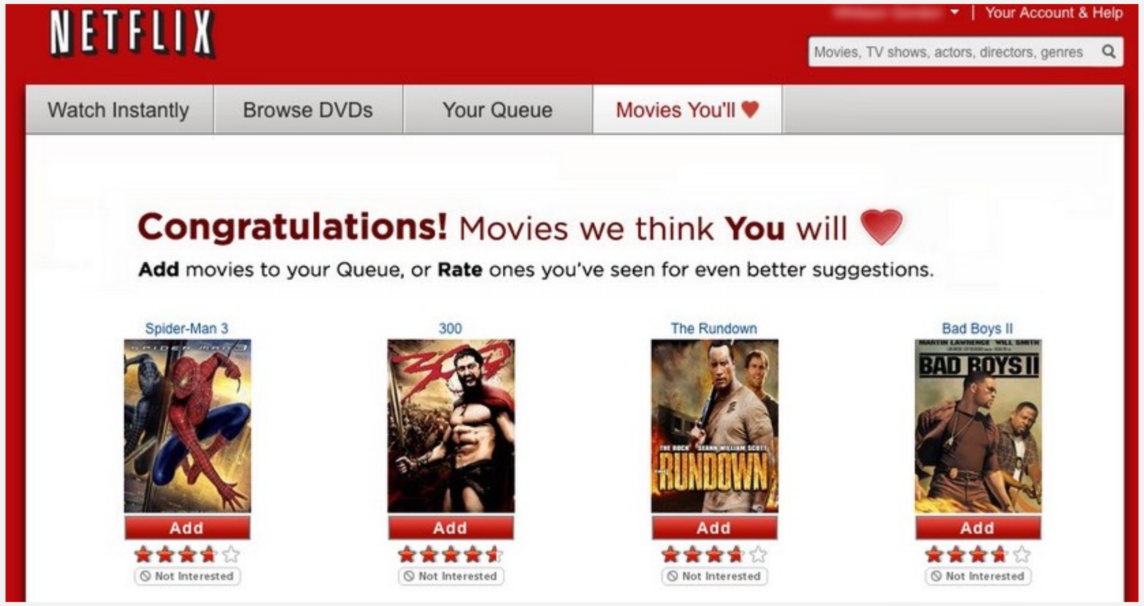
Deep nets fail to learn semantics





Constructive preference elicitation

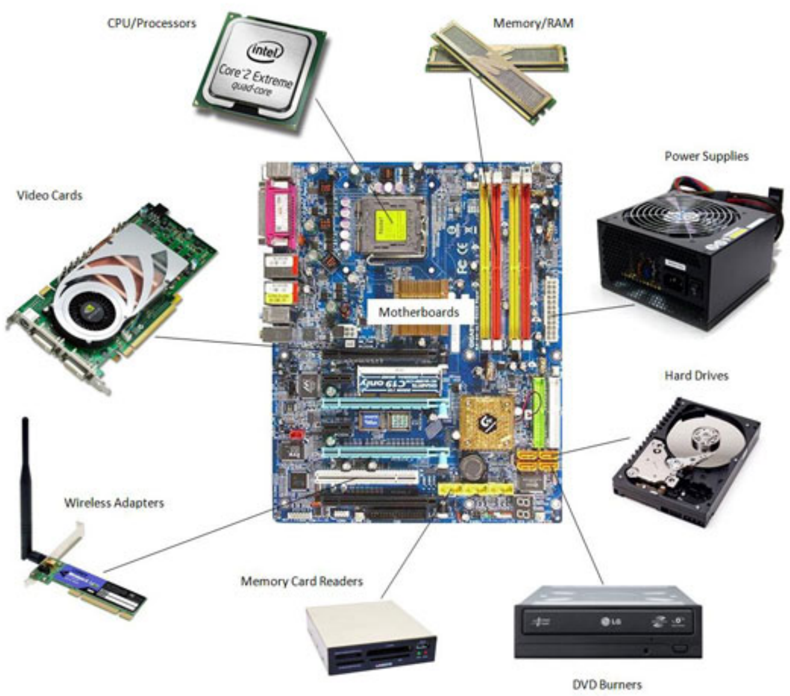
- Preference elicitation ***selects the most preferred item*** in a catalogue of candidates



- Constructive preference elicitation aims at **synthesising the most preferred configuration** given a set of constraints

Examples: assembly

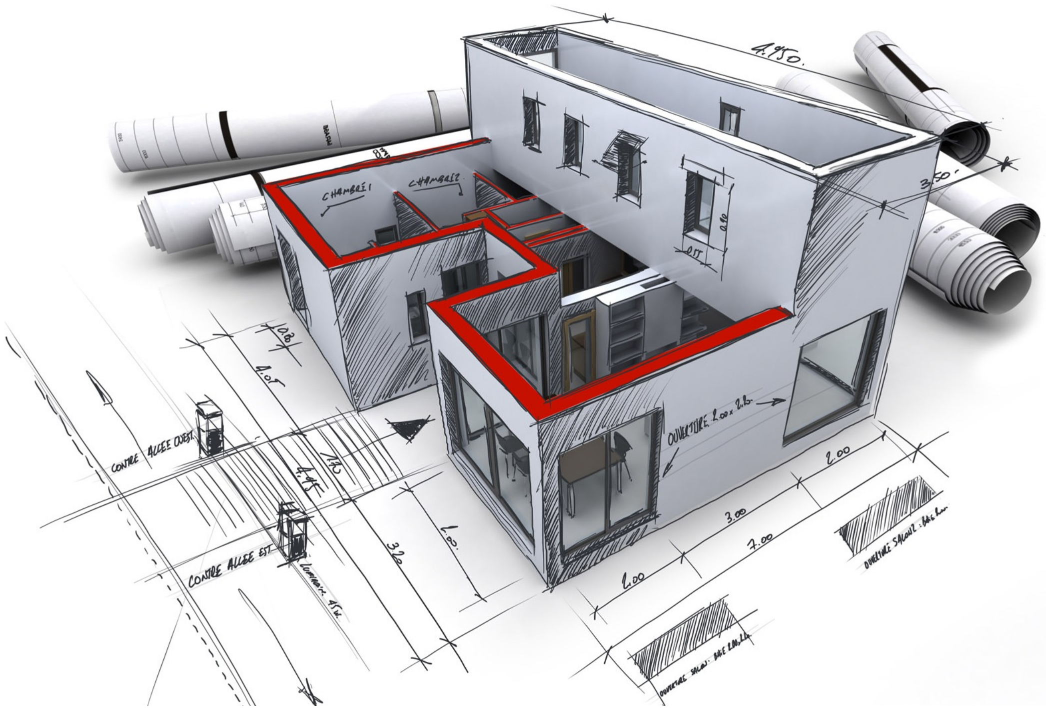
compose a basket



assemble a PC

modify a recipe

Examples: layout synthesis



building design

interior design

urban planning

Examples: the ultimate cocktail machine!

Plain preference elicitation

Daily Driver

Off-road

Muscle Car

Collectable

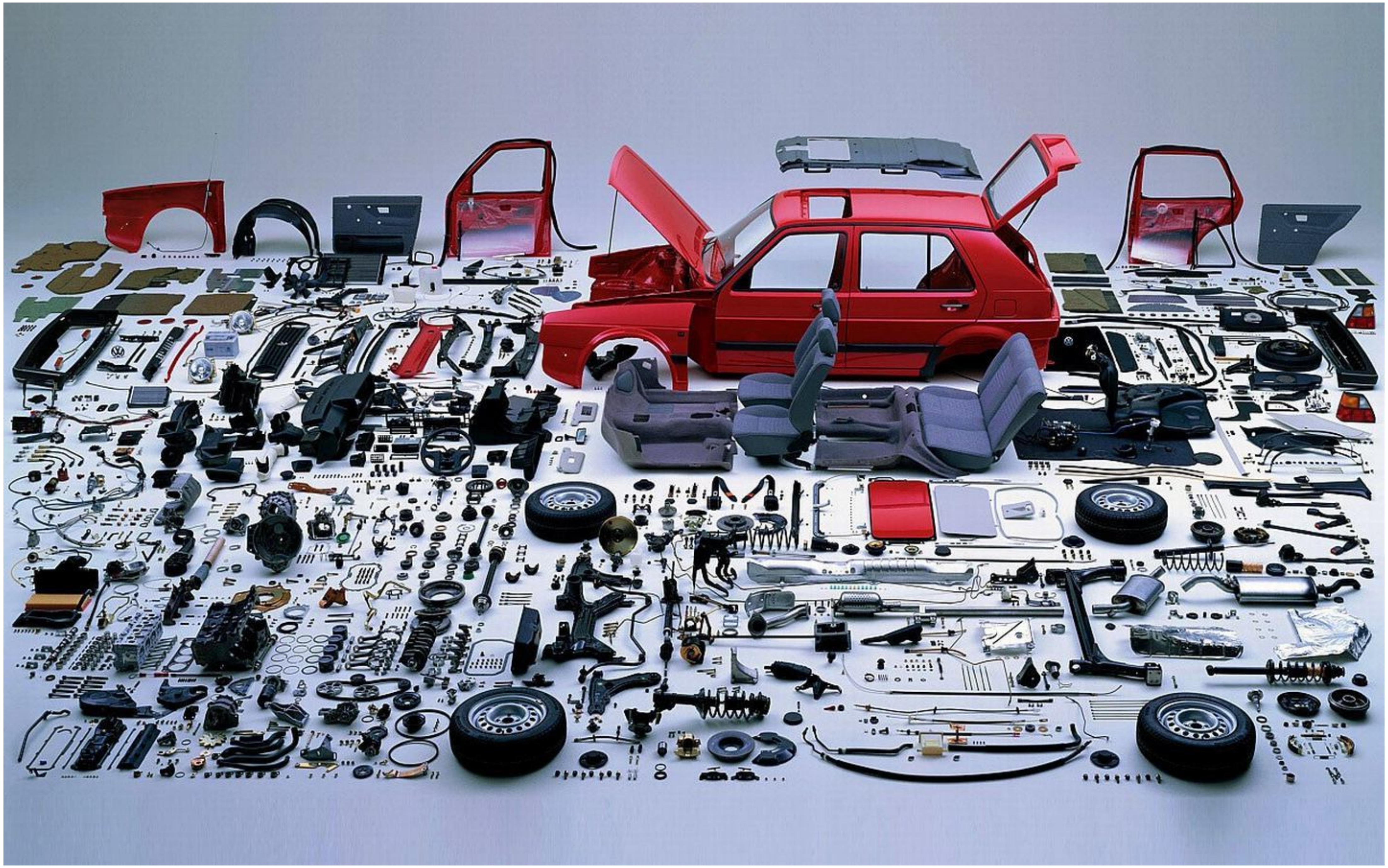
Luxury

Wagon

Track

Hauler

Constructive preference elicitation



learning user utilities

- **User utility** as weighted combination of item features

weights · given context

$$U(x,y) = \langle \boldsymbol{w}, \boldsymbol{\varphi}(x,y) \rangle$$

feature function · candidate output

- Recommendation as **utility maximisation**

$$y^* = \operatorname*{argmax}_{y \in \mathcal{Y}_{\text{feasible}}} \langle \boldsymbol{w}, \boldsymbol{\varphi}(x,y) \rangle$$

feasibility constraints

- **Interactive approaches** to learn utility weights

Leveraging **proactive users**

[ECMLPKDD 2018, RECSYS 2018]

PROBLEM

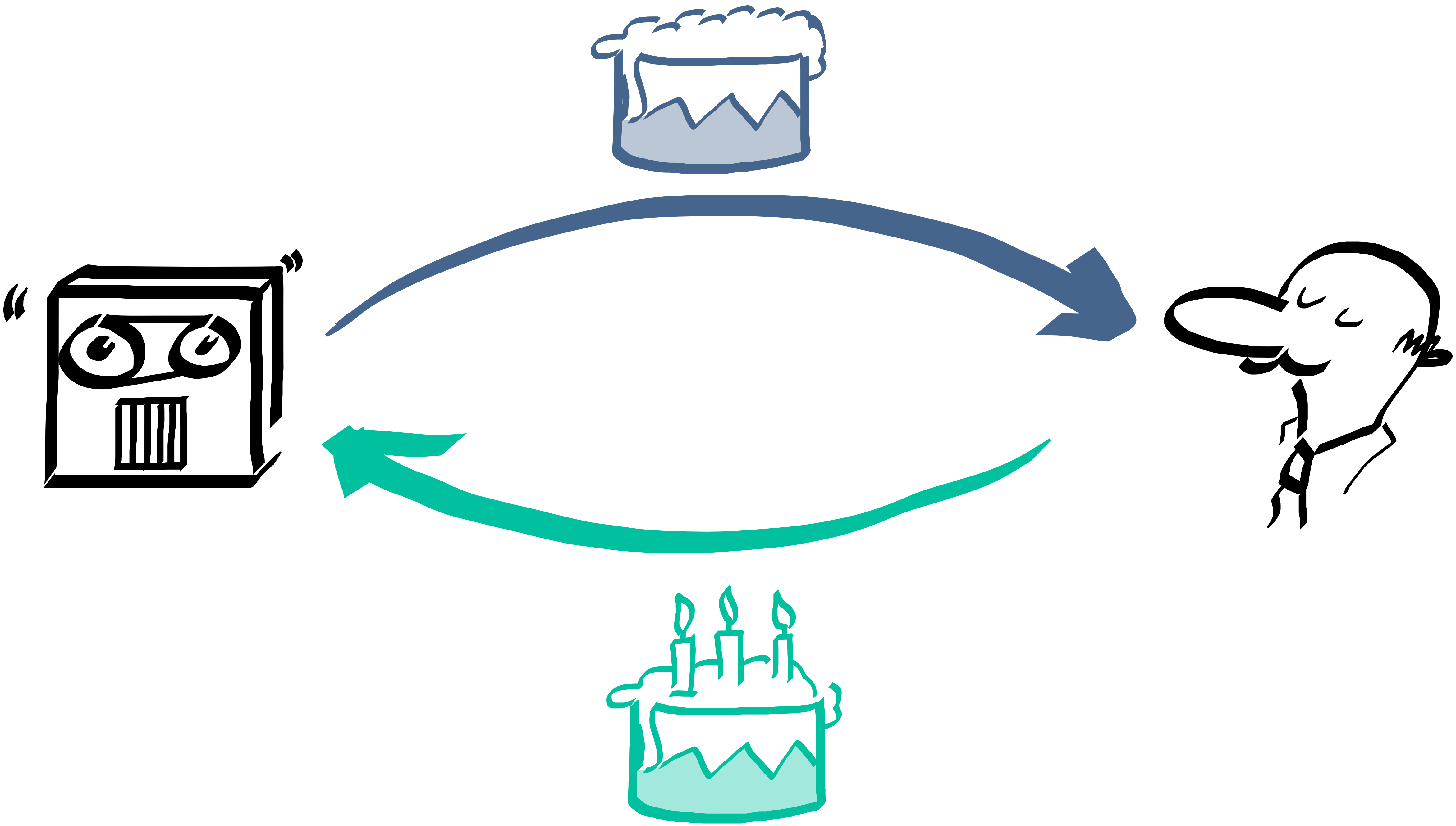
- Most interactive approaches ask for **pairwise (or setwise) preferences**
- The set of **candidates** is **predetermined**

SOLUTION

- Ask user to improve current solution **(coactive learning)**
- Improvements **over features or objects**

Coactive learning

(Shivaswamy & Joachims, 2012)



Coactive learning (CL)

procedure $\mathrm{CL}(T)$

$\boldsymbol{w}^1 \leftarrow 0$

for $t = 1, \ldots, T$ **do**

Receive context x^t from the user

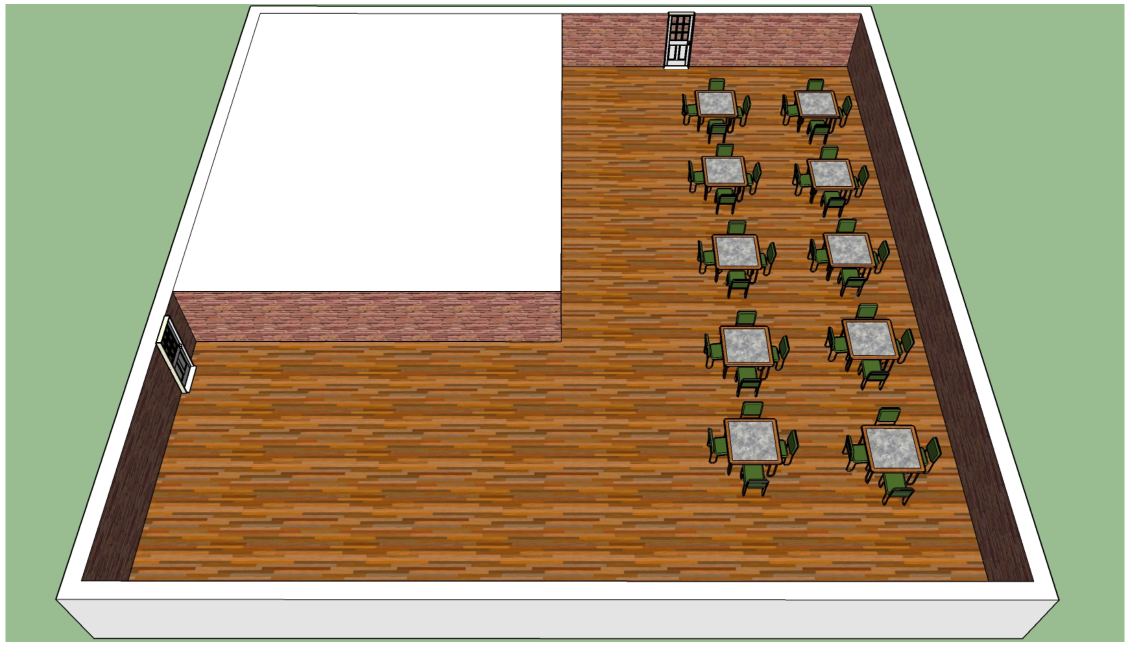
$y^t \leftarrow \operatorname{argmax}_{y \in \mathcal{Y}} \langle \boldsymbol{w}^t, \boldsymbol{\varphi}(x^t, y) \rangle$

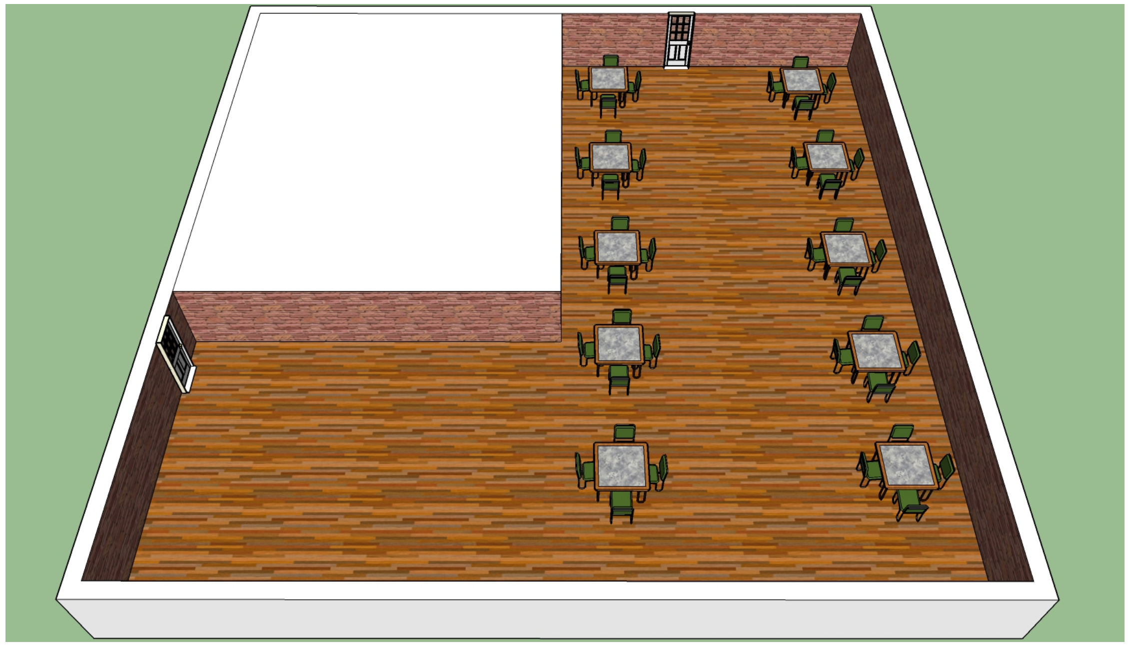
$\bar{y}^t \leftarrow \textsc{QueryImprovement}(x^t, y^t)$

$\boldsymbol{w}^{t+1} \leftarrow \boldsymbol{w}^t + \boldsymbol{\varphi}(x^t, \bar{y}^t) - \boldsymbol{\varphi}(x^t, y^t)$

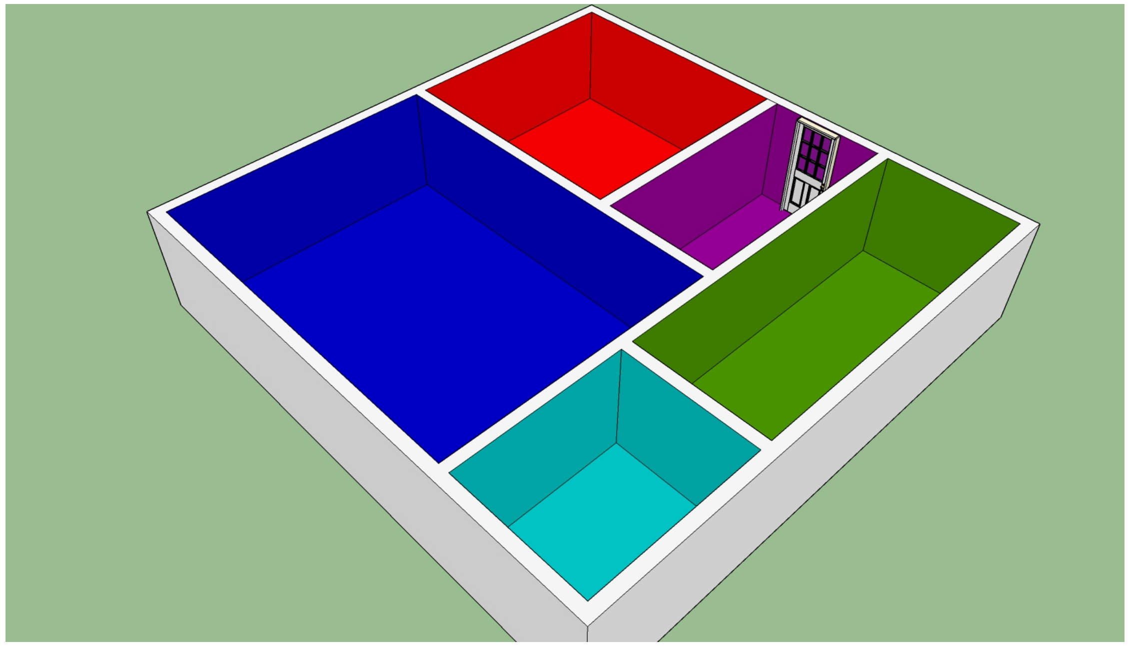
Layout synthesis: user interaction

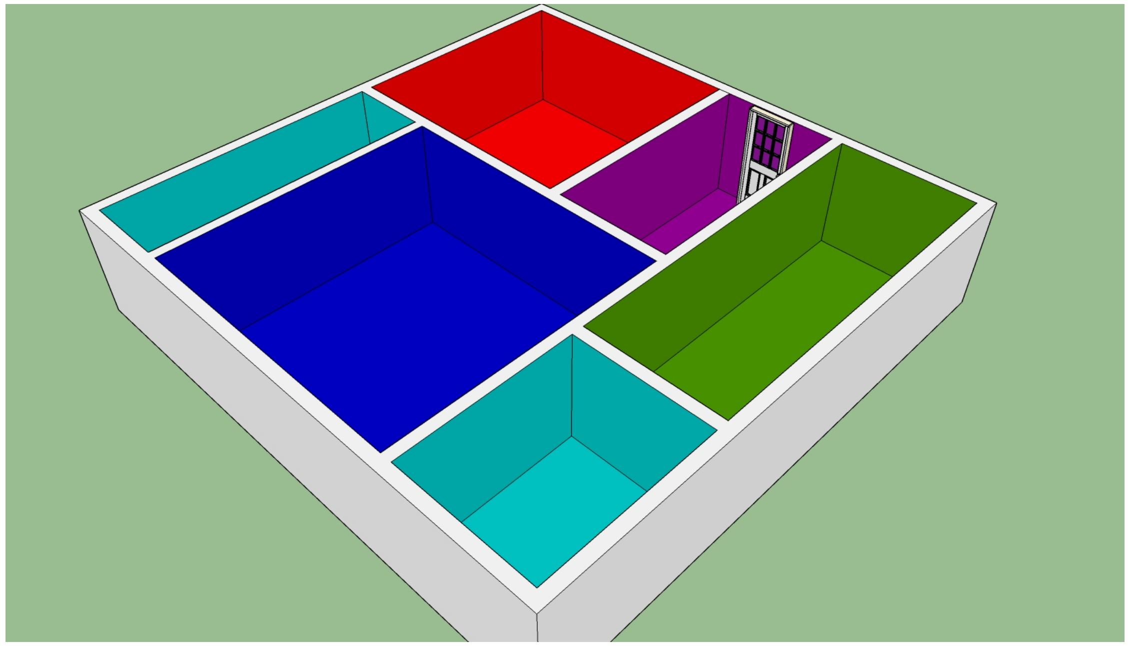
feature manipulation



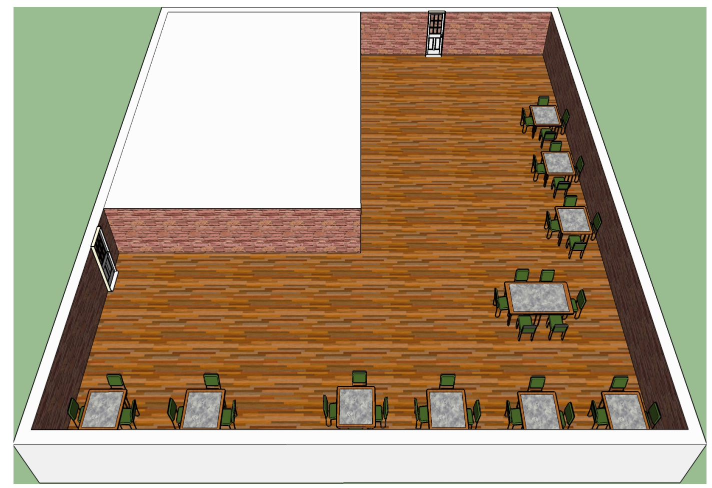
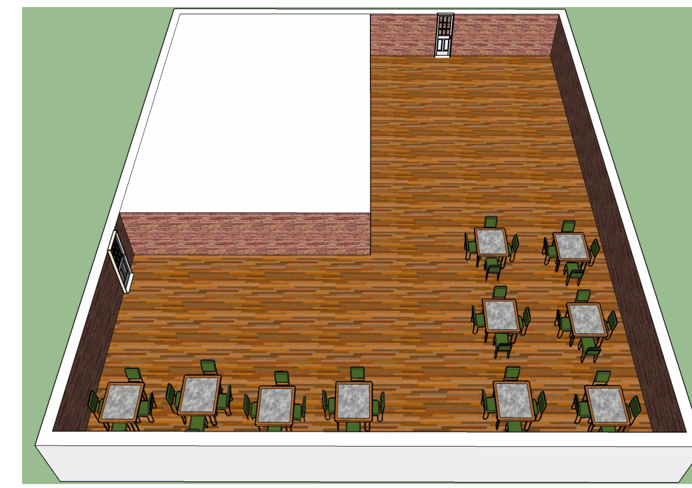
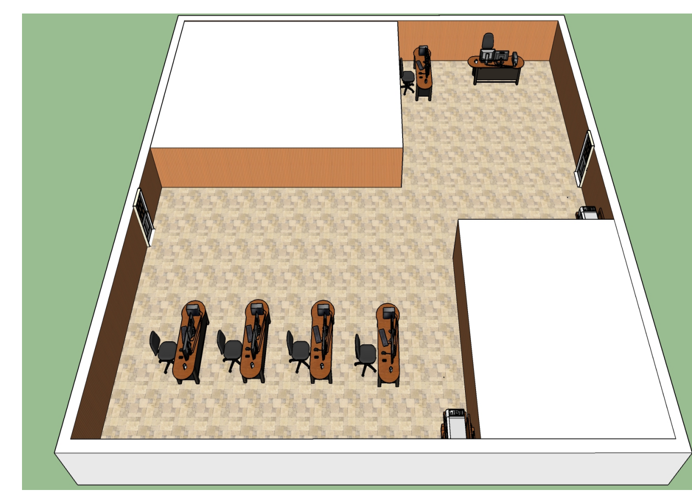


object manipulation

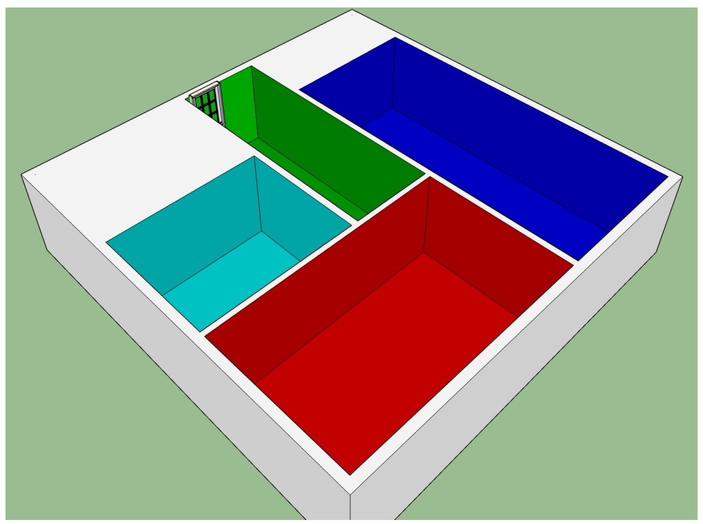
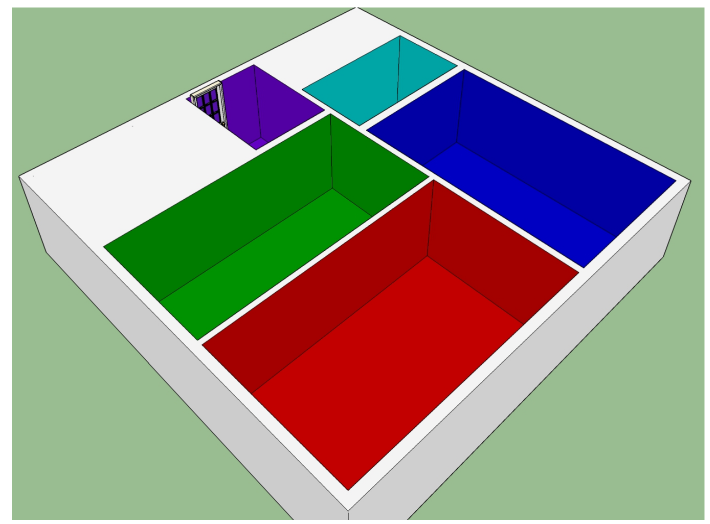
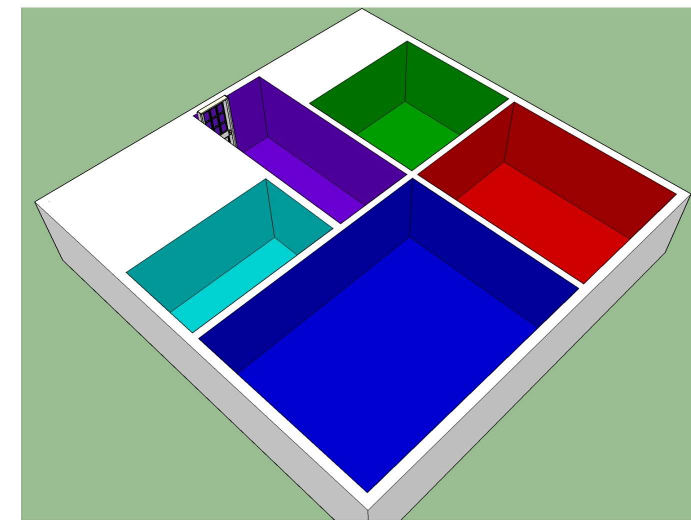
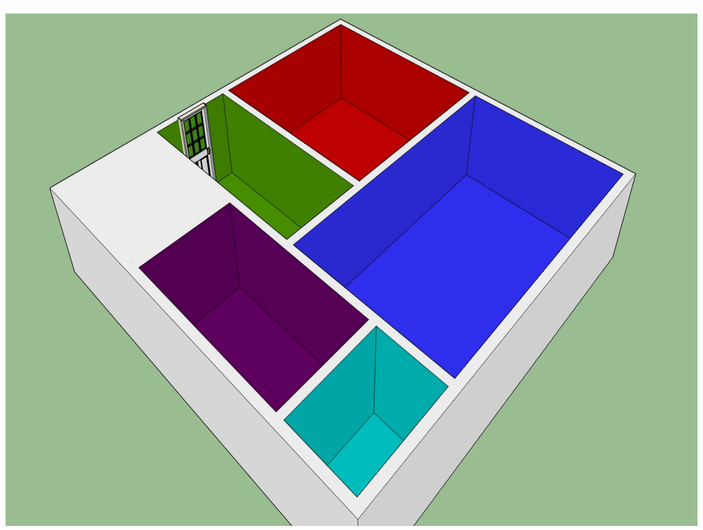
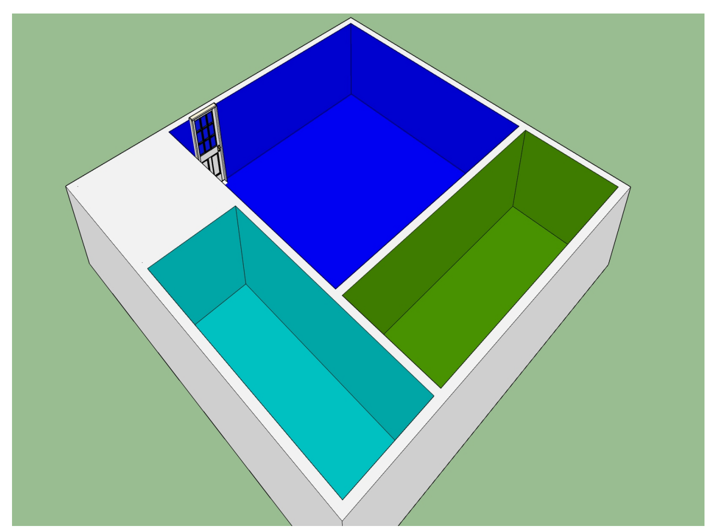
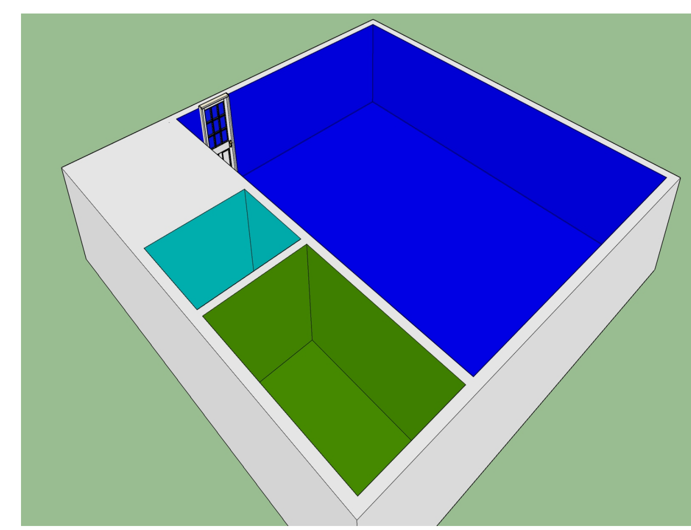




Layout synthesis: furniture arrangement

	Initial	Intermediate	Final
Café			
Office			

Layout synthesis: floor planning

	Initial	Intermediate	Final
Flat			
Loft			

Leveraging **user explanations**

[AAAI, 2017]

PROBLEM

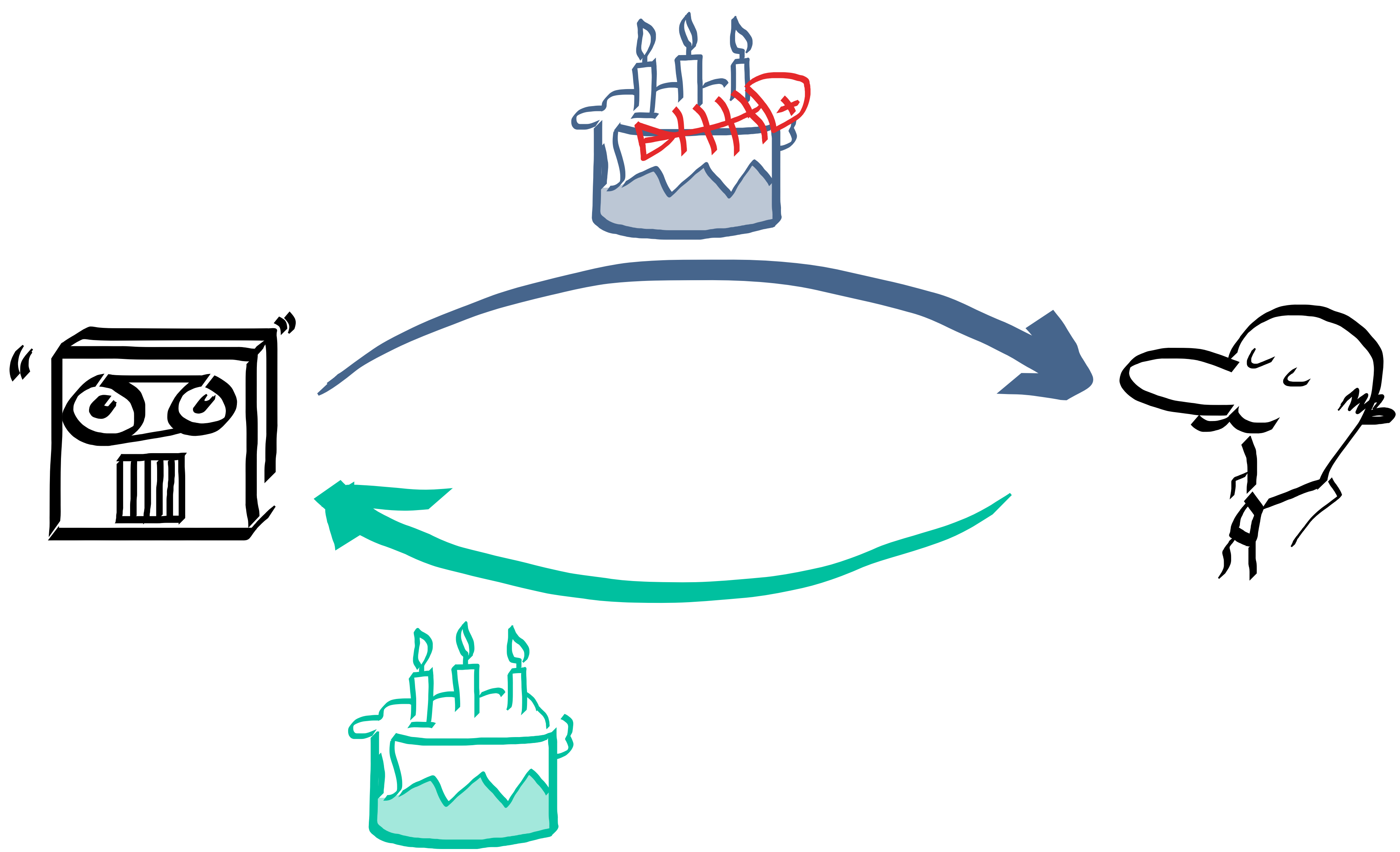
- Most interactive learning approaches assume **predefined feature space**
- Users often **realise requirements during process**

SOLUTION

- Identify when **feature space is insufficient**
- Ask user to explain improvement **(coactive critiquing)**

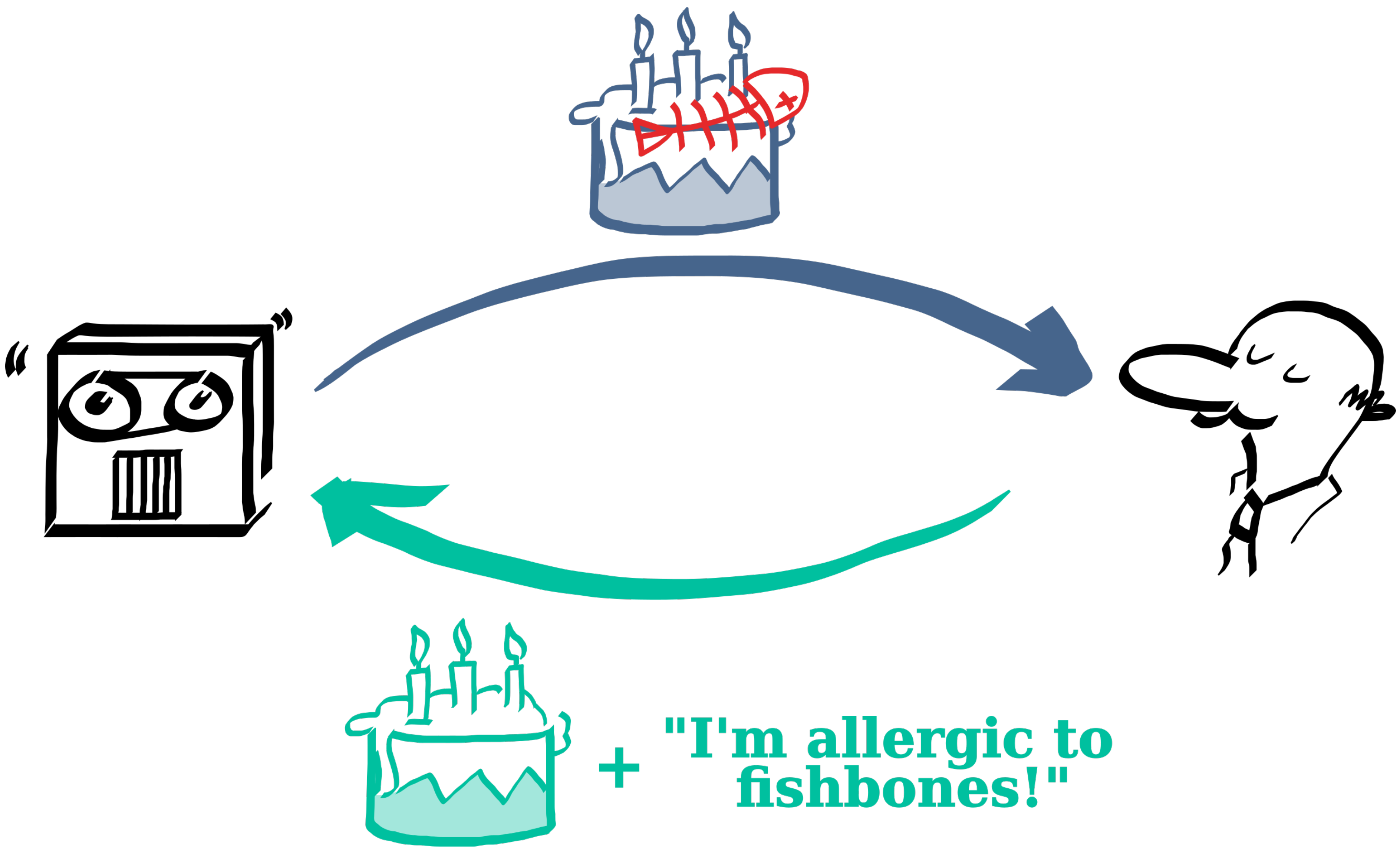
Solution: coactive critiquing

eliciting local improvements (**coactive learning**)



Solution: coactive critiquing

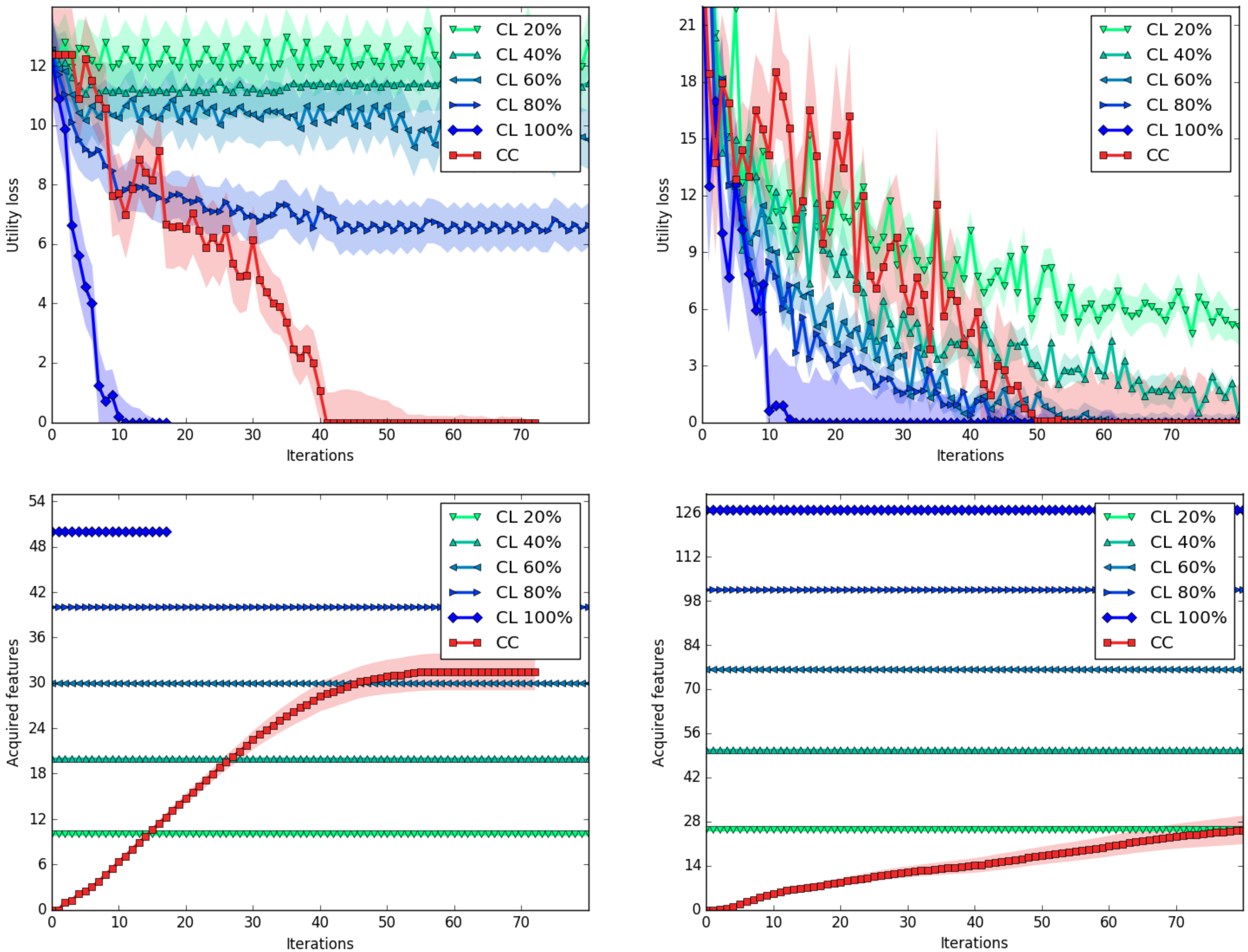
asking for explanations if needed (**critiquing**)



Coactive critiquing algorithm (CC)

$$
\begin{aligned}
&\textbf{procedure } \mathrm{CC}(\boldsymbol{\varphi}^1, T)\\
&\quad \boldsymbol{w}^1 \leftarrow 0, \;\; \mathcal{D} \leftarrow \emptyset\\
&\quad \textbf{for } t = 1, \ldots, T \textbf{ do}\\
&\qquad \text{Receive context } x^t \text{ from the user}\\
&\qquad y^t \leftarrow \mathrm{argmax}_{y \in \mathcal{Y}} \langle \boldsymbol{w}^t, \boldsymbol{\varphi}^t(x^t, y) \rangle\\
&\qquad \bar{y}^t \leftarrow \textsc{QueryImprovement}(x^t, y^t)\\
&\qquad \mathcal{D} \leftarrow \mathcal{D} \cup \{(x^t, y^t, \bar{y}^t)\}\\
&\qquad \textbf{if } \textsc{NeedCritique}(\mathcal{D}, \boldsymbol{\varphi}^t) \textbf{ then}\\
&\qquad\quad \rho \leftarrow \textsc{QueryCritique}(x^t, y^t, \bar{y}^t)\\
&\qquad\quad \boldsymbol{\varphi}^t \leftarrow \boldsymbol{\varphi}^t \circ [\rho]\\
&\qquad\quad \boldsymbol{w}^t \leftarrow \boldsymbol{w}^t \circ [0]\\
&\qquad \boldsymbol{w}^{t+1} \leftarrow \boldsymbol{w}^t + \boldsymbol{\varphi}^t(x^t, \bar{y}^t) - \boldsymbol{\varphi}^t(x^t, y^t)\\
&\qquad \boldsymbol{\varphi}^{t+1} \leftarrow \boldsymbol{\varphi}^t
\end{aligned}
$$

Experimental results: CC vs CL



Going deep:
Dealing with **unknown features**

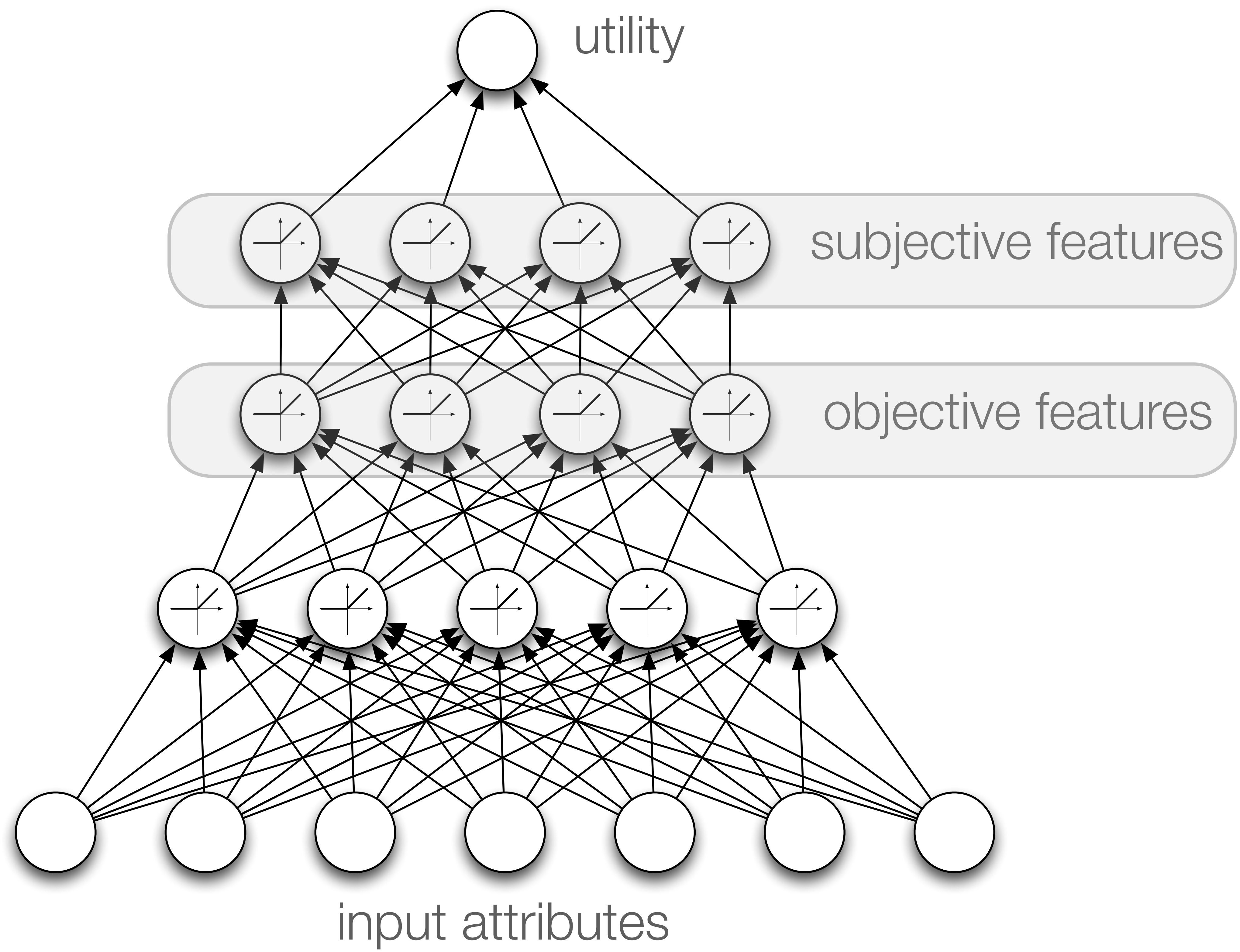
PROBLEM

- Most preference elicitation approaches assume **predefined feature space**
- It is often **difficult** for users to provide **explicit features**

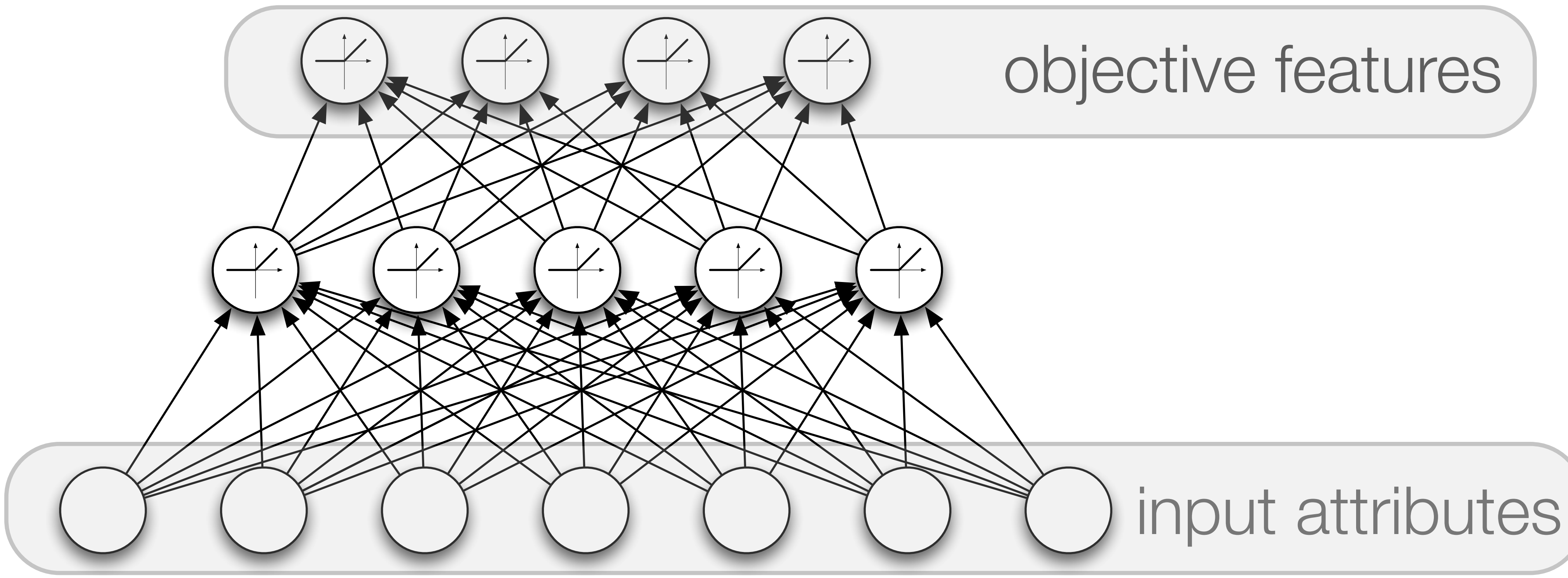
SOLUTION

- Learn features in coactive mode (**deep coactive learning**)

ReLU Neural Networks

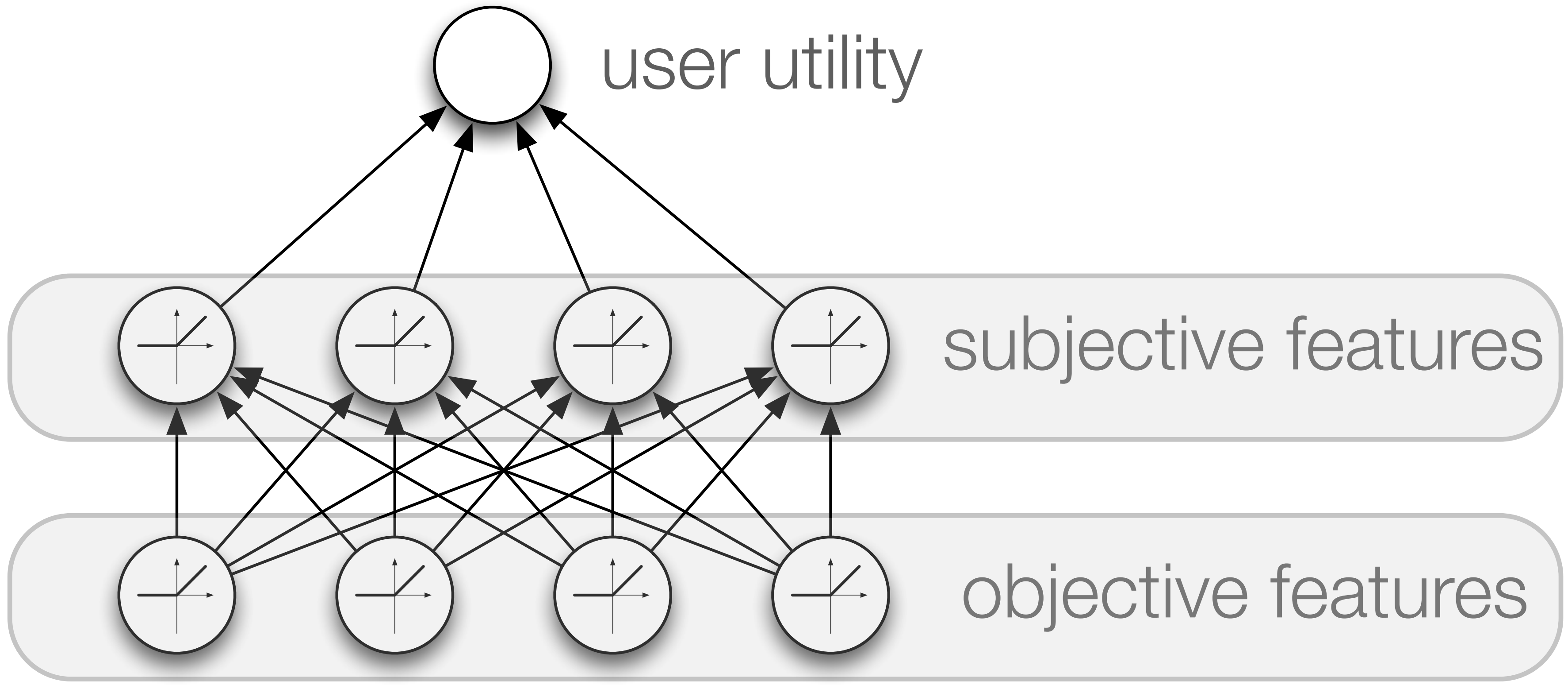


Learn objective features



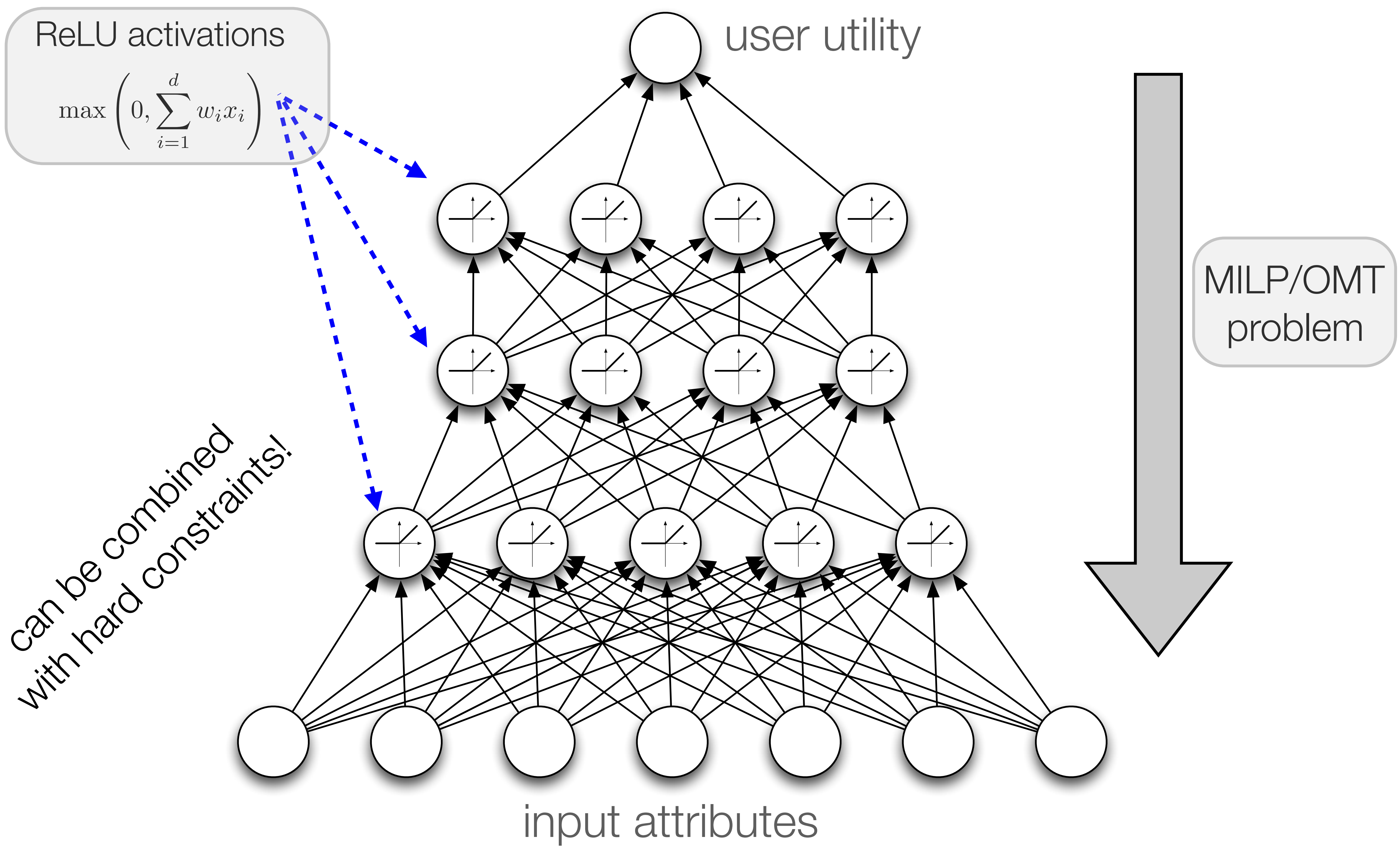
- Train a NN to predict **"objective" features** from attributes (e.g. sweet, spicy)
- Train with **feedback** from **multiple users** (i.e. "objective" as average over users)

From objective features to user utility

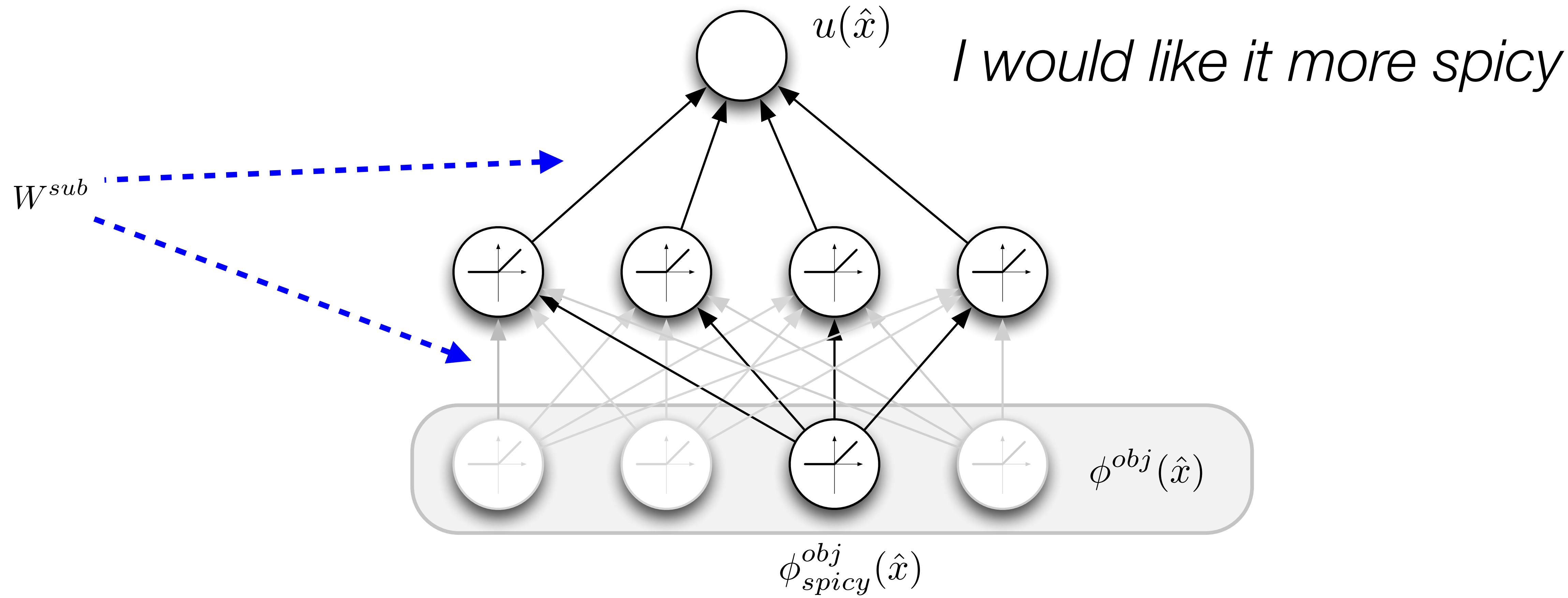


- Compose with user-specific component:
    - from **objective** to **subjective** features
    - from **subjective** features to **utility** score (preference)

Optimize over input to get recommendation



Coactive feedback backpropagation



$$W^{sub} = W^{sub} + \eta \nabla_{W^{sub}} \left(\frac{\partial\, u(x)}{\partial\, \phi^{obj}_{spicy}(x)} \right)_{x=\hat{x}}$$

Credits

P. Viappiani
(CNRS - Lip6)

M. Kumar
(KULEUVEN)

S. Teso
(KULEUVEN)

M. Vescovi
(Telecom)

K. Tentori
(CIMEC - UNITN)

R. Sebastiani
(DISI - UNITN)

P. Campigotto
(TU Dortmund)

P. Dragone
(DISI - UNITN)

G. Pellegrini
(DISI - UNITN)

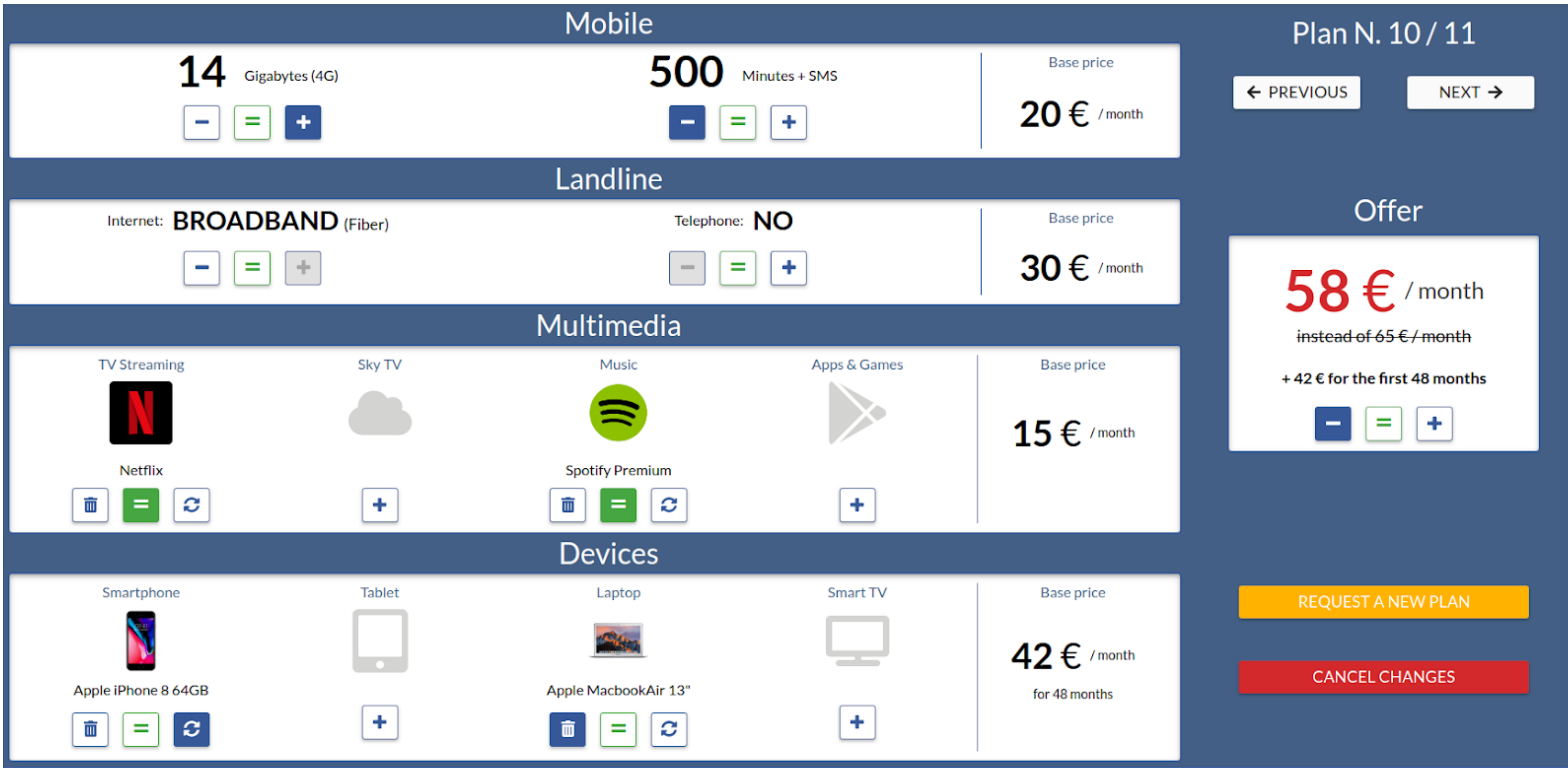
L. Erculiani
(DISI - UNITN)

Thank you, questions?

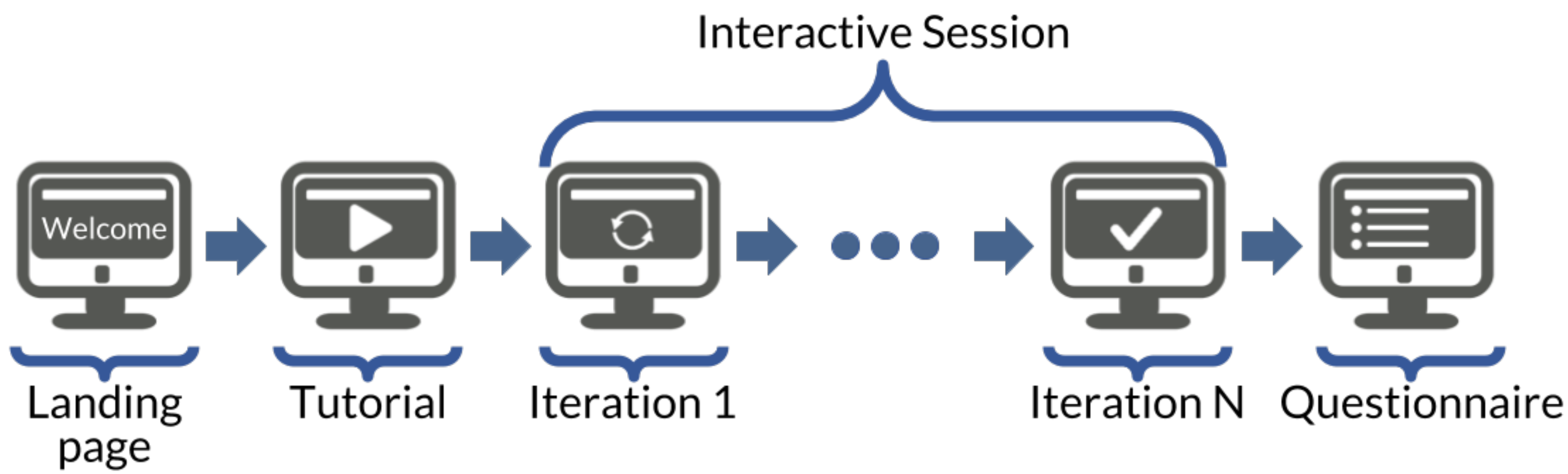
- .A. Passerini, **Learning Modulo Theories,** Springer, Cham, 2016
- S. Teso, A. Passerini, P. Viappiani, **Constructive Preference Elicitation by Setwise Max-Margin Learning**, IJCAI 2016.
- S. Teso, R. Sebastiani, A. Passerini, **Structured Learning Modulo Theories**, Artificial Intelligence Journal, 2017 .
- S. Teso, P. Dragone, A. Passerini, **Coactive Critiquing: Elicitation of Preferences and Features**, AAAI, 2017.
- P. Dragone, S. Teso, A. Passerini, **Constructive Preference Elicitation**, Frontiers in Robotics and AI, 2018.
- P. Dragone, S. Teso, A. Passerini, **Constructive Preference Elicitation over Hybrid Combinatorial Spaces**, AAAI 2018.
- P. Dragone, S. Teso, M. Kumar, A. Passerini, **Decomposition Strategies for Constructive Preference Elicitation**, AAAI 2018.
- P. Dragone, S. Teso, A. Passerini, **Pyconstruct: Constraint Programming Meets Structured Prediction**, IJCAI 2018.
- L. Erculiani, P. Dragone, S. Teso, A. Passerini, **Automating Layout Synthesis with Constructive Preference Elicitation**, ECMLPKDD 2018.
- P. Dragone, G. Pellegrini, M. Vescovi, K. Tentori, A. Passerini, **No more ready-made deals: constructive recommendation for telco service bundling**, RECSYS 2018.

Smart Plan: navigation

Smart Plan: suggesting changes



Smart Plan: experimental setup

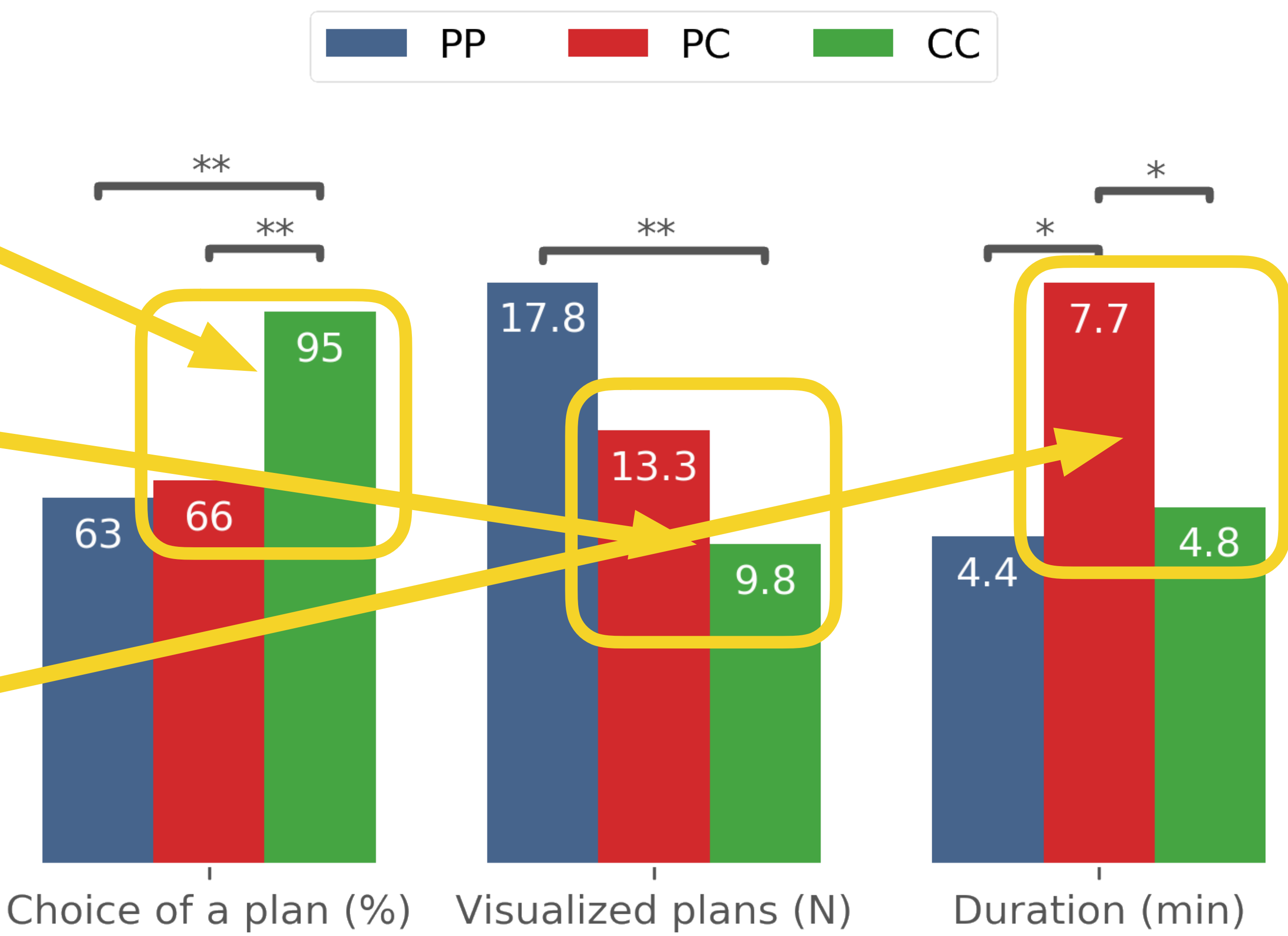


System	Interface	Algorithm
CC	Constructive	Constructive
CP	Constructive	Pool
PP	Pool	Pool

- **Between groups** experiment
- Comparison **against pool-based** approaches
- **Pool algorithm**: algorithm can choose from (large) pool of options
- **Pool interface**: user can choose from (small) pool of option

Smart Plan: quantitative results

- 29% more satisfactory interactions
- 3.5 interactions less to satisfactory plan
- 2.9 minutes less to satisfactory plan



Preference elicitation example: housing

I would like a house in a safe area, close to my parents and the kindergarten, with a garden if there are no parks nearby. My maximum budget is 300,000 euro.

MAX-SMT formulation

var	description	var	description
y_1	has garden	y_2	has park nearby
y_3	crime rate	y_4	distance from parents
y_5	distance from kindergarten		

$$\max_{\mathbf{x}} \quad w_1\psi_1 + w_2\psi_2 + w_3(\psi_3 \wedge \psi_4)$$

client utility

subject to:

client soft constraints

$$\begin{aligned} \psi_1 &= (\neg y_2 \Rightarrow y_1) \\ \psi_2 &= (y_3 \leq \theta_1) \\ \psi_3 &= (y_4 \leq \theta_2) \\ \psi_4 &= (y_5 \leq \theta_3) \end{aligned}$$

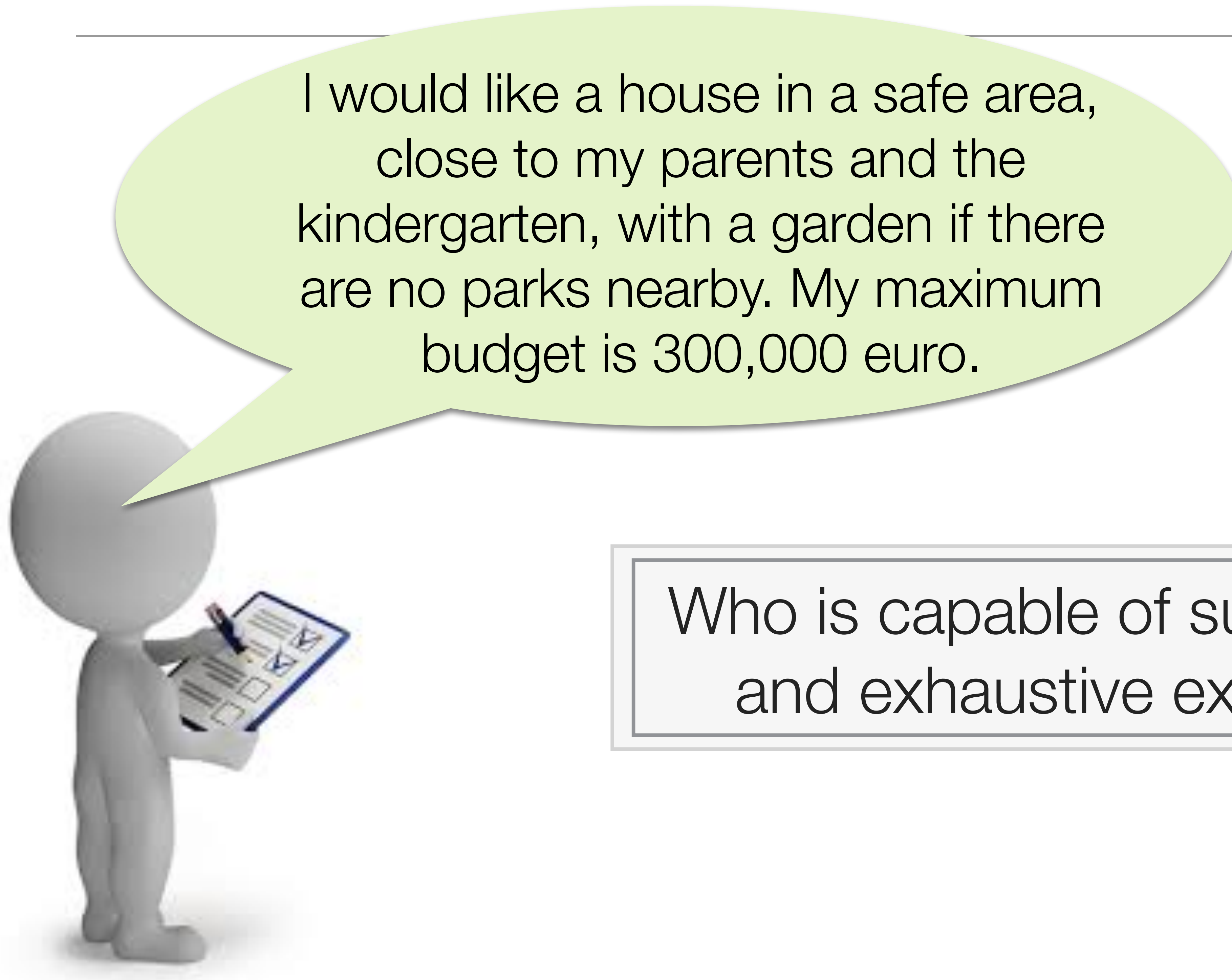
client hard constraints

$$price(\mathbf{y}) \leq 300000$$

$$y_4 \geq \theta_4$$
$$y_5 \geq \theta_5$$

company hard constraints

housing revisited



Who is capable of such a precise and exhaustive explanation?

Solving an unknown MAT-SMT problem

- exact problem **formulation unknown**
- set of candidate **catalogue features** is available
- set of **candidate constraints** over the features
- true (unknown) utility is the weighted sum of **few constraints over few features**
- DM feedback as **pairwise preferences** btw solutions

Problem formulation: catalogue features

set of features characterising candidate solutions

feature	Description	type
y_1	house type	ord
y_2	garden	Bool
y_3	garage	Bool
y_4	commercial facilities nearby	Bool
y_5	public green areas nearby	Bool
y_6	distance from downtown	num
y_7	crime rate	num
y_9	public transit service quality index	num
y_{10}	distance from parents house	num
....	...	...

Problem formulation: possible predicates

all predicates constructible with candidate features

predicate	Description	formula
p_1	has garden	y_2
p_2	has garage	y_3
p_3	has park nearby	y_5
p_4	close to downtown	$y_6 < \theta_1$
p_5	low crime rate area	$y_7 < \theta_2$
p_6	high quality transit service	$y_8 > \theta_3$
....	...	...

Problem formulation: possible constraints

all constraints constructible with candidate predicates
(combinations of up to d predicates)

constraint	Description	formula
ψ_1	has garden	p_1
ψ_2	garden if no park nearby	$\neg p_3 \rightarrow p_1$
ψ_3	good transportation if far from downtown	$\neg p_4 \rightarrow p_6$
ψ_4	garage if high crime rate	$\neg p_5 \rightarrow p_2$
....	...	...

User utility

- (Unknown) user utility linear combination of **some** constraints
- Scoring function linear combination of **all** candidate constraints

$$\mathbf{y}^* = \mathrm{argmax}_{\mathbf{y}} \mathbf{w}^T \boldsymbol{\varphi}(\mathbf{y})$$

- Need to learn the **few non-zero** weights
- User feedback as **pairwise preference** btw candidate solutions

Learning user utility: learning to rank

sparsifying term

penalty term

$$\min_{\mathbf{w},\xi} \quad ||\mathbf{w}||_1 + \lambda \sum_{i,j:\mathbf{y}_i \prec \mathbf{y}_j} \xi_{i,j}$$

penalty for not satisfying constraint

subject to:

$$\mathbf{w}^T \boldsymbol{\varphi}(\mathbf{y}_i) - \mathbf{w}^T \boldsymbol{\varphi}(\mathbf{y}_j) \geq 1 - \xi_{i,j}$$

$$\forall i, j : \mathbf{y}_i \prec \mathbf{y}_j$$

minimal distance

match user pairwise preferences

Learning algorithm

- Initialise weights
- While user not satisfied
  - run **MAX-SMT** to find candidate configurations:

$$\mathbf{y}^* = \mathrm{argmax}_{\mathbf{y}} \mathbf{w}^T \boldsymbol{\varphi}(\mathbf{y})$$

  - collect feedback as pairwise preferences:

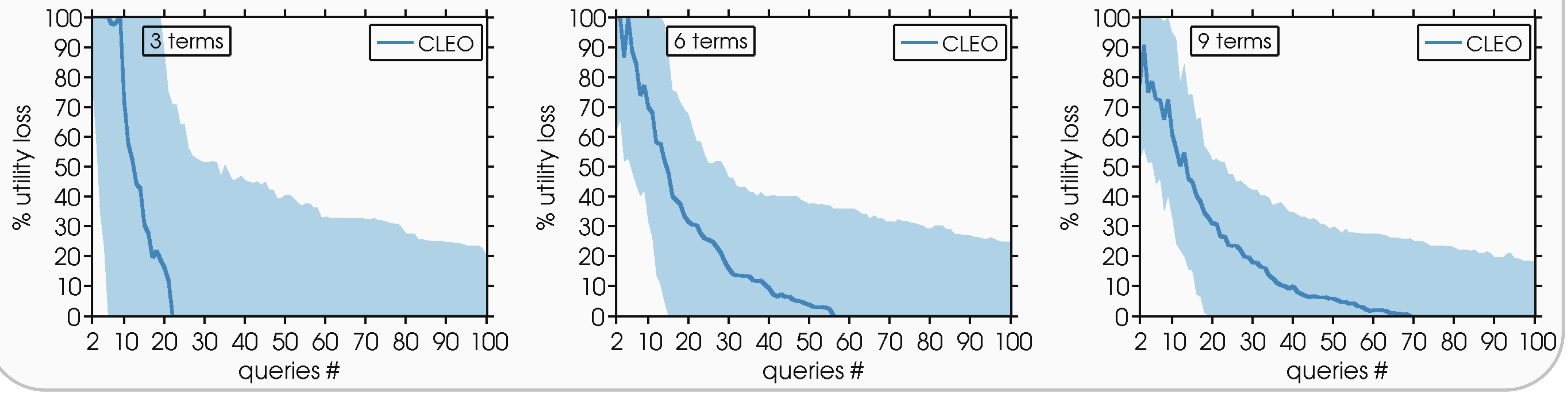
$$\mathbf{y}_i \prec \mathbf{y}_j$$

  - add constraints and solve learning problem:

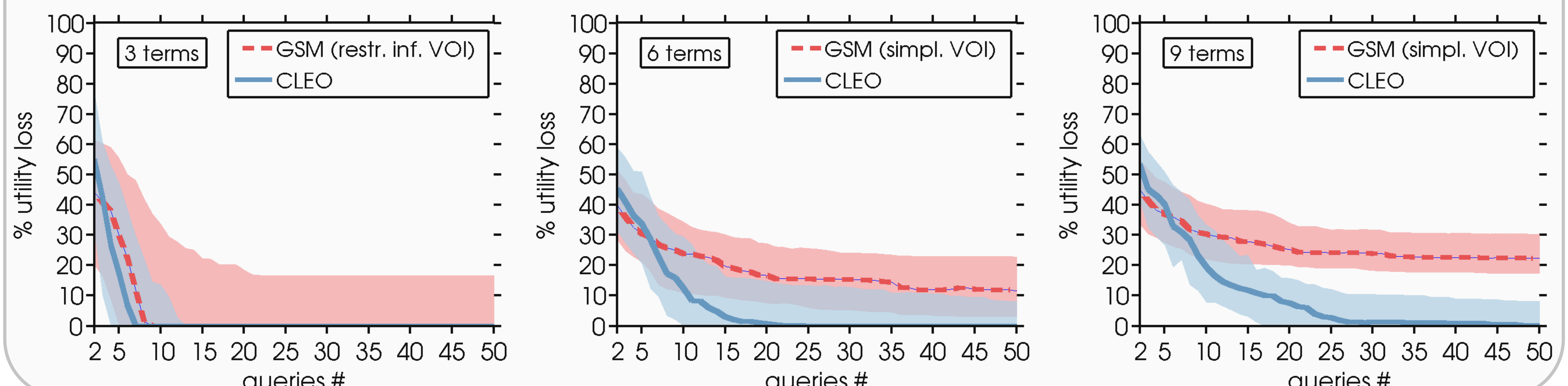
$$\min_{\mathbf{w},\xi} \qquad ||\mathbf{w}||_1 + \lambda \sum_{i,j:\mathbf{y}_i \prec \mathbf{y}_j} \xi_{i,j}$$

Experimental results

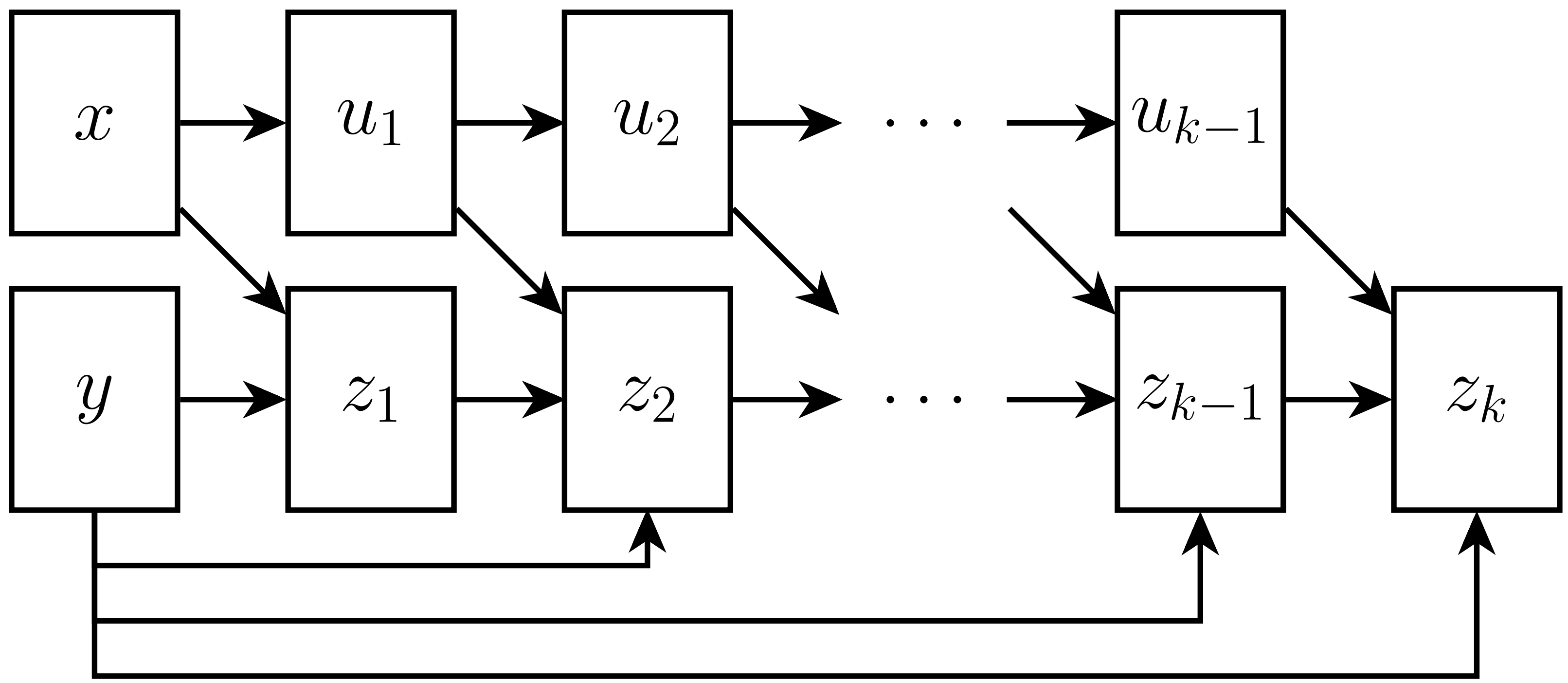
HOUSING



BOOLEAN



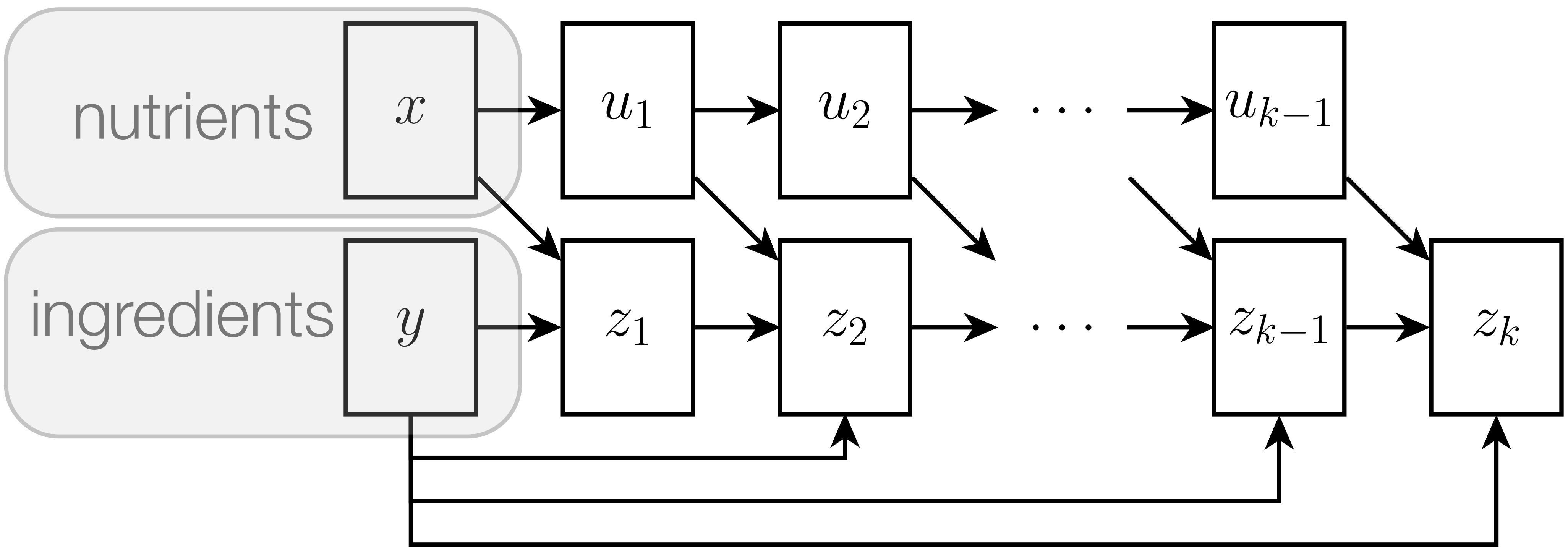
Partially Input Convex Neural Nets (PICNN)



- Requirements:
  - **convex and non-decreasing** activations functions
  - **non-negative** weights in z layers
- Results:
  - network is **convex in y**

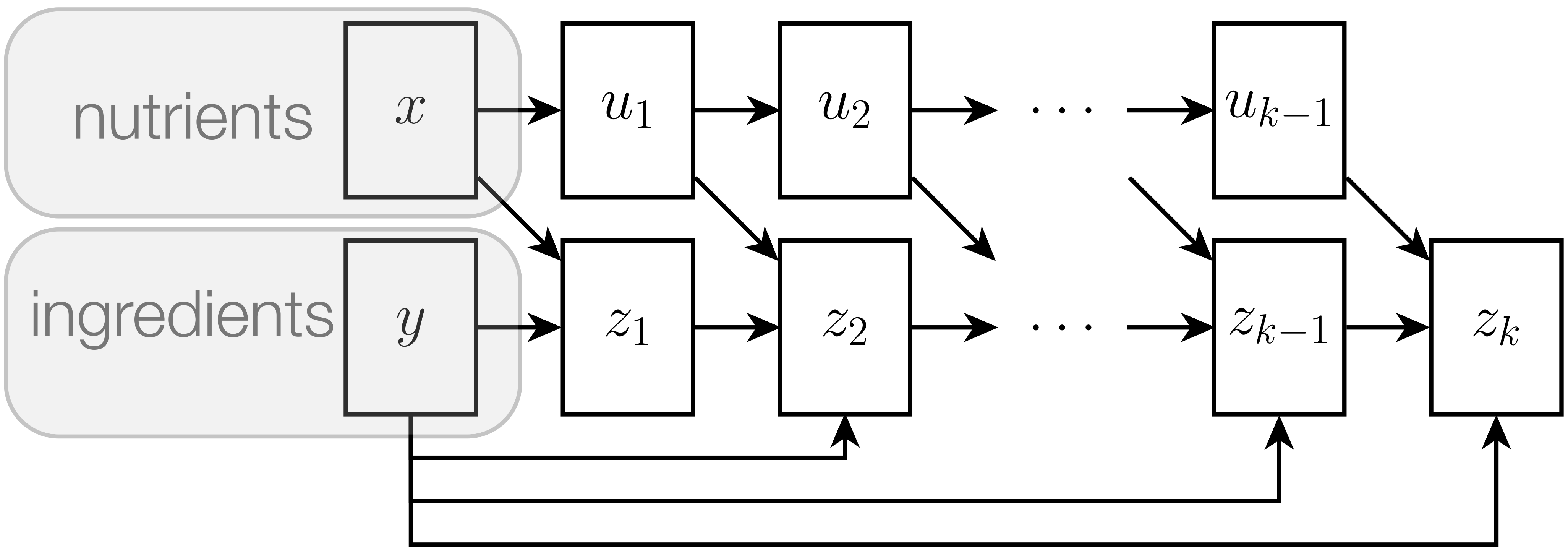
$$\hat{y} = \min_{y \in \mathcal{Y}} f(\hat{x}, y)$$

PICNN for constructive recommendation



- Train PICNN to predict **ingredients from nutrients**

PICNN for constructive recommendation



- Recommend new product by:
  - given a nutrients-ingredients pair and **desired nutrients**

$$\langle \hat{x}, \hat{y} \rangle \;\rightarrow\; \langle \hat{x}', ? \rangle$$

  - get **minimal ingredient change** giving the desired nutrients

$$\hat{y}' = \min_{y \in \mathcal{Y}} f(\hat{x}', y) + \lambda\, \ell(y, \hat{y})$$