Deep Learning with Hard Logical Constraints

Eleonora Giunchiglia
TU Wien
Institute of Logics and Computation

Outline

1. Introduction
2. Hierarchical Constraints [1]
3. Normal Logic Constraints [2]
4. A Novel Benchmark for Neuro-symbolic Models [3]
5. Open Questions
6. Q&A

[1] Giunchiglia E., Lukasiewicz T., Coherent Hierarchical Multi-label Classification Networks, NeurIPS, 2020
[2] Giunchiglia E., Lukasiewicz T., Multi-label Classification Neural Networks with Hard Logical Constraints, JAIR, 2021
[3] Giunchiglia E., Stoian M., Khan S., Cuzzolin F., Lukasiewicz T., ROAD-R: The Autonomous Driving Dataset with Logical Requirements,MLJ, 2022

Introduction

Deep neural networks have been responsible for SOTA for many years, however they can exhibit unexpected behaviours and make mistakes that an “intelligent being” would never commit.

Introduction

Deep neural networks have been responsible for SOTA for many years, however they can exhibit unexpected behaviours and make mistakes that an “intelligent being” would never commit.

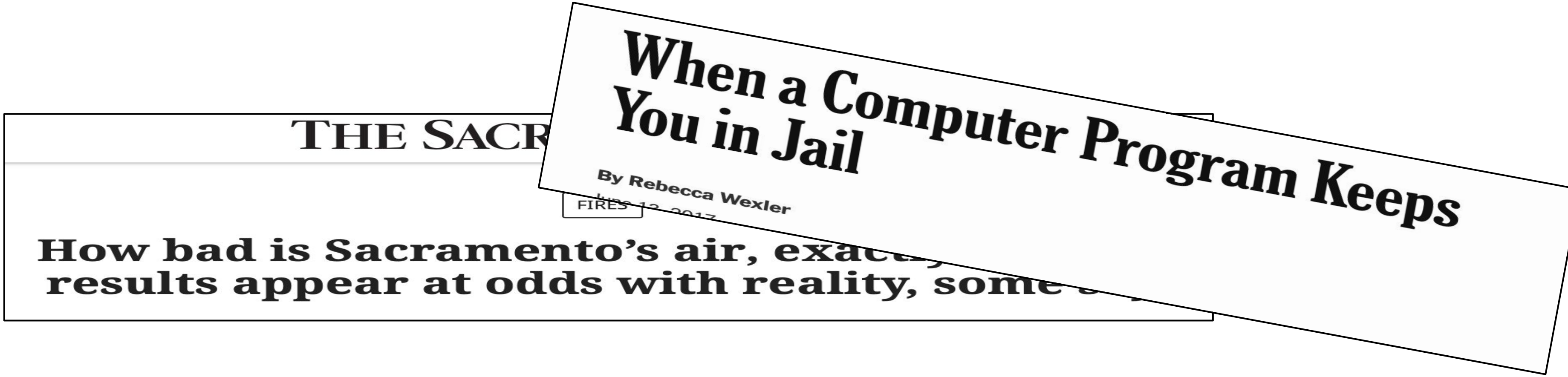
THE SACRAMENTO BEE

FIRES

How bad is Sacramento’s air, exactly? Google results appear at odds with reality, some say

Introduction

Deep neural networks have been responsible for SOTA for many years, however they can exhibit unexpected behaviours and make mistakes that an “intelligent being” would never commit.



Introduction

Deep neural networks have been responsible for SOTA for many years, however they can exhibit unexpected behaviours and make mistakes that an “intelligent being” would never commit.

Introduction

Is it possible to build neural networks that:

- are guaranteed to be compliant with the constraints, and
- can learn from the constraints themselves?

Introduction

Is it possible to build neural networks that:

- are guaranteed to be compliant with the constraints, and
- can learn from the constraints themselves?

*

* with some assumptions about the syntax of the constraints

Introduction

Focus on **multi-label classification problems**

Why?

1. It is possible to assign a human-understandable semantics to the outputs

2. There often exist correlations among outputs

3. We often have background knowledge about such correlations

We can write hard logical constraints that define the admissible output space of the developed models

Outline

1. Introduction
2. Hierarchical Constraints [1]
3. Normal Logic Constraints [2]
4. A Novel Benchmark for Neuro-symbolic Models [3]
5. Summary and Open Questions
6. Q&A

[1] Giunchiglia E., Lukasiewicz T., Coherent Hierarchical Multi-label Classification Networks, NeurIPS, 2020
[2] Giunchiglia E., Lukasiewicz T., Multi-label Classification Neural Networks with Hard Logical Constraints, JAIR, 2021
[3] Giunchiglia E., Stoian M., Khan S., Cuzzolin F., Lukasiewicz T., ROAD-R: The Autonomous Driving Dataset with Logical Requirements,MLJ, 2022

Hierarchical Constraints

Two classes: A, B

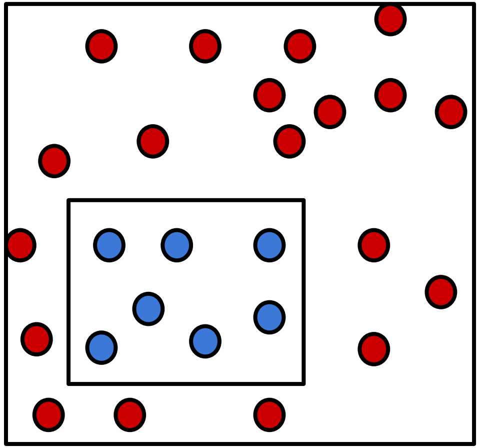
Constraint: $A \to B$

An Introductory Example

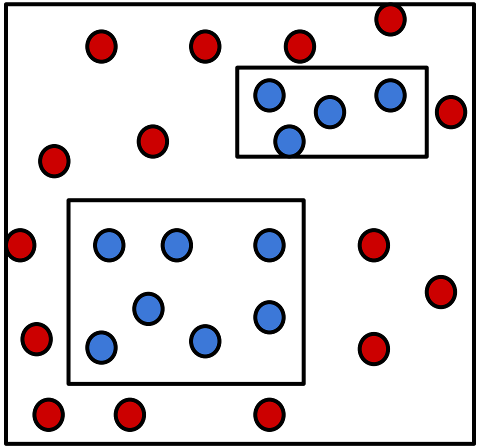
Two classes: A, B

Constraint: $A \rightarrow B$

Class A



Class B

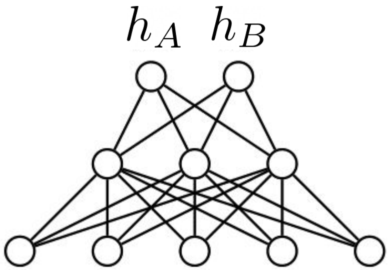


Standard Solution

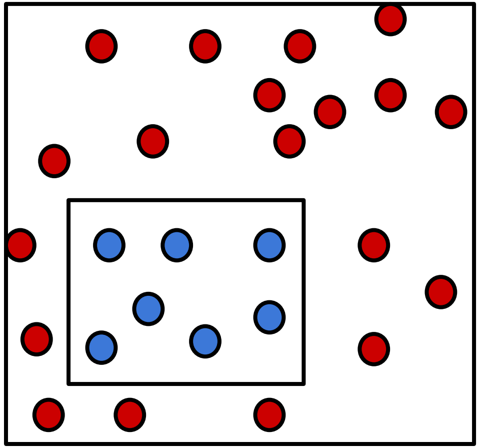
Two classes: A, B

Constraint: $A \rightarrow B$

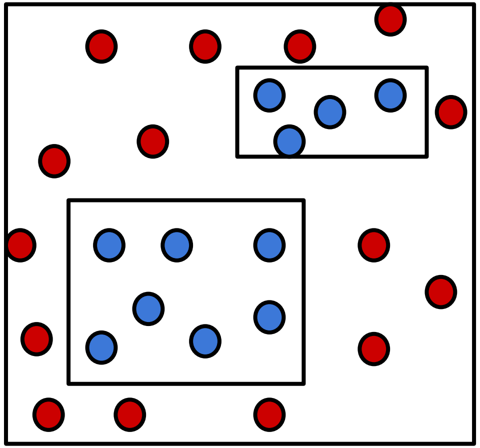
Standard Approach:



Class A



Class B

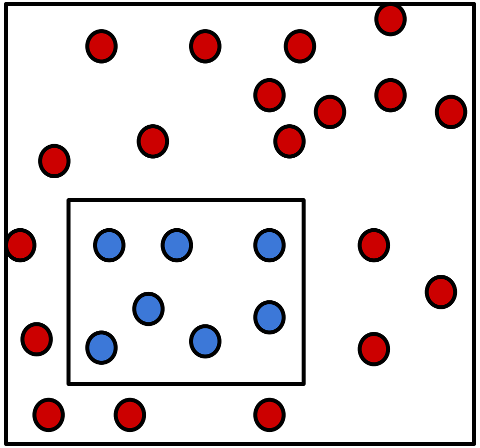


Standard Solution

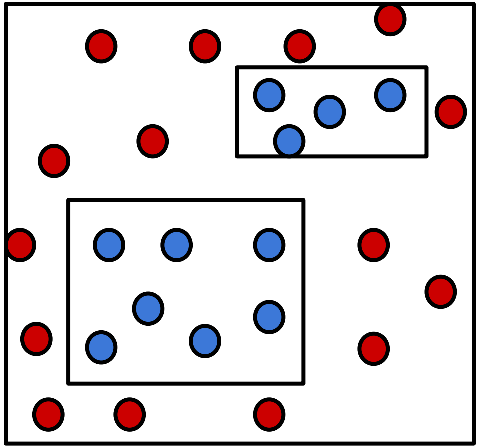
Two classes: A, B

Constraint: $A \to B$

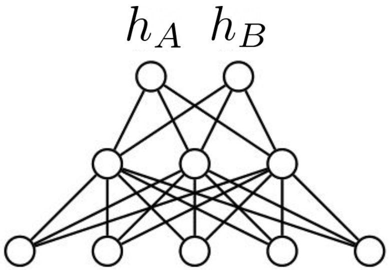
Class A



Class B



Standard Approach:



Problem:

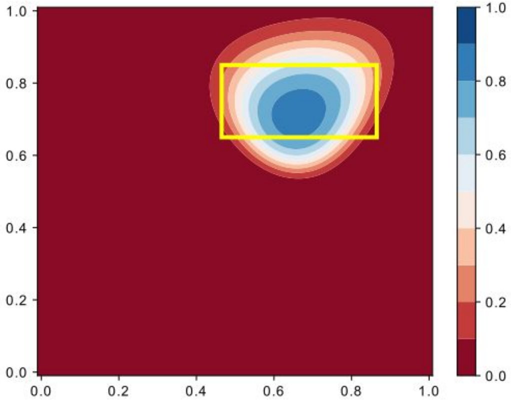
1. no guarantees to satisfy $A \to B$
2. no exploitation of hierarchical knowledge

Hierarchical Constraints

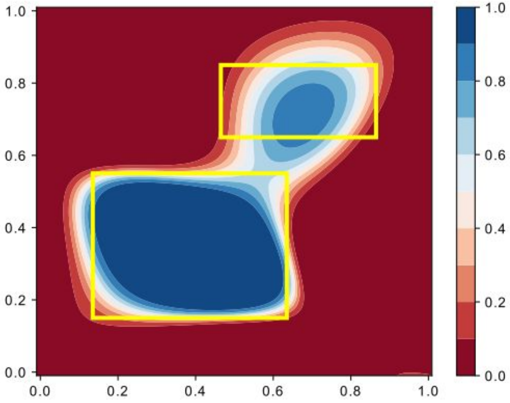
Two classes: A, B

Constraint: $A \to B$

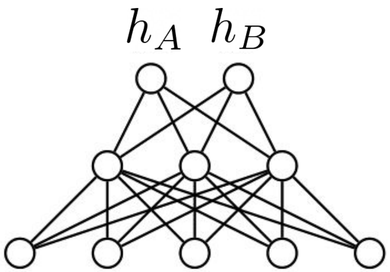
Class A



Class B



Standard Approach:

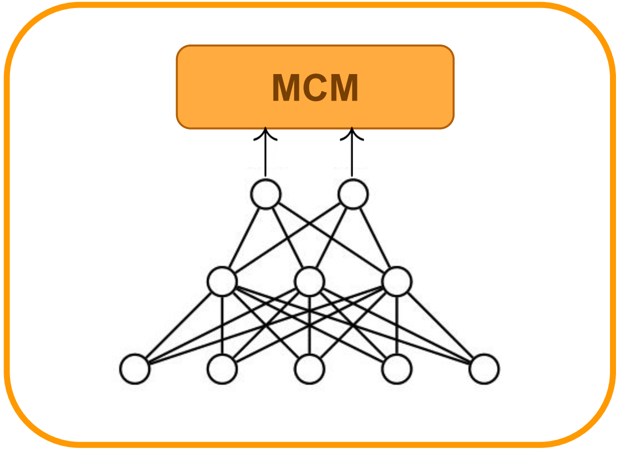


We implemented a simple neural network with one hidden layer and 7 neurons

Hierarchical Constraints: Our Solution

Build a Max Constraint Module (MCM) on top of the standard neural network

$$\mathrm{MCM}_A = h_A,$$
$$\mathrm{MCM}_B = \max(h_B, h_A).$$



Our solution

Build a Max Constraint Module on top of the standard neural network

$$\mathrm{MCM}_A = h_A,$$
$$\mathrm{MCM}_B = \max(h_B, h_A).$$

MCM ensures the satisfaction of the constraint

Our solution

Build a Max Constraint Module on top of the standard neural network

$$\mathrm{MCM}_A = h_A,$$
$$\mathrm{MCM}_B = \max(h_B, h_A).$$

MCM ensures the satisfaction of the constraint

MCM can cause wrong supervisions

Our solution

Build a Max Constraint Module on top of the standard neural network

$$\text{MCM}_A = h_A,$$
$$\text{MCM}_B = \max(h_B, h_A).$$

MCM ensures the satisfaction of the constraint

MCM can cause wrong supervisions

⚠ If $h_A > h_B, y_A = 0, y_B = 1$ then we teach y_A to increase instead of decreasing

Our solution

Build a Max Constraint Module on top of the standard neural network

$$\text{MCM}_A = h_A,$$
$$\text{MCM}_B = \max(h_B, h_A).$$

MCM ensures the satisfaction of the constraint

MCM can cause wrong supervisions

⚠ If $h_A > h_B, y_A = 0, y_B = 1$ then we teach y_A to increase instead of decreasing

$$\text{MCLoss}_A = -y_A \ln(\text{MCM}_A) - (1 - y_A)\ln(1 - \text{MCM}_A),$$
$$\text{MCLoss}_B = -y_B \ln(\max(h_B, h_A y_A)) - (1 - y_B)\ln(1 - \text{MCM}_B)).$$

Our solution

Build a Max Constraint Module on top of the standard neural network

$$\mathrm{MCM}_A = h_A,$$
$$\mathrm{MCM}_B = \max(h_B, h_A).$$

MCM ensures the satisfaction of the constraint

MCM can cause wrong supervisions

If $h_A > h_B, y_A = 0, y_B = 1$ then we teach y_A to increase instead of decreasing

$$\mathrm{MCLoss}_A = -y_A \ln(\mathrm{MCM}_A) - (1 - y_A)\ln(1 - \mathrm{MCM}_A),$$
$$\mathrm{MCLoss}_B = -y_B \ln(\max(h_B, h_A y_A)) - (1 - y_B)\ln(1 - \mathrm{MCM}_B)).$$

Our solution

Build a Max Constraint Module on top of the standard neural network

$$\mathrm{MCM}_A = h_A,$$
$$\mathrm{MCM}_B = \max(h_B, h_A).$$

MCM ensures the satisfaction of the constraint

MCM can cause wrong supervisions

If $h_A > h_B, y_A = 0, y_B = 1$ then we teach y_A to increase instead of decreasing

$$\mathrm{MCLoss}_A = -y_A \ln(\mathrm{MCM}_A) - (1 - y_A)\ln(1 - \mathrm{MCM}_A),$$
$$\mathrm{MCLoss}_B = -y_B \ln(\max(h_B, h_A y_A)) - (1 - y_B)\ln(1 - \mathrm{MCM}_B)).$$

Our solution

Build a Max Constraint Module on top of the standard neural network

$$\mathrm{MCM}_A = h_A,$$
$$\mathrm{MCM}_B = \max(h_B, h_A).$$

MCM ensures the satisfaction of the constraint

MCM can cause wrong supervisions

⚠ If $h_A > h_B, y_A = 0, y_B = 1$ then we teach y_A to increase instead of decreasing

$$\mathrm{MCLoss}_A = -y_A \ln(\mathrm{MCM}_A) - (1 - y_A)\ln(1 - \mathrm{MCM}_A),$$
$$\mathrm{MCLoss}_B = -y_B \ln(\max(h_B, h_A y_A)) - (1 - y_B)\ln(1 - \mathrm{MCM}_B)).$$

Our solution

Build a Max Constraint Module on top of the standard neural network

$$\text{MCM}_A = h_A,$$
$$\text{MCM}_B = \max(h_B, h_A).$$

MCM ensures the satisfaction of the constraint

MCM can cause wrong supervisions

If $h_A > h_B, y_A = 0, y_B = 1$ then we teach y_A to increase instead of decreasing

$$\text{MCLoss}_A = -y_A \ln(\text{MCM}_A) - (1 - y_A)\ln(1 - \text{MCM}_A),$$
$$\text{MCLoss}_B = \boxed{-y_B \ln(\max(h_B, h_A y_A))} - (1 - y_B)\ln(1 - \text{MCM}_B)).$$

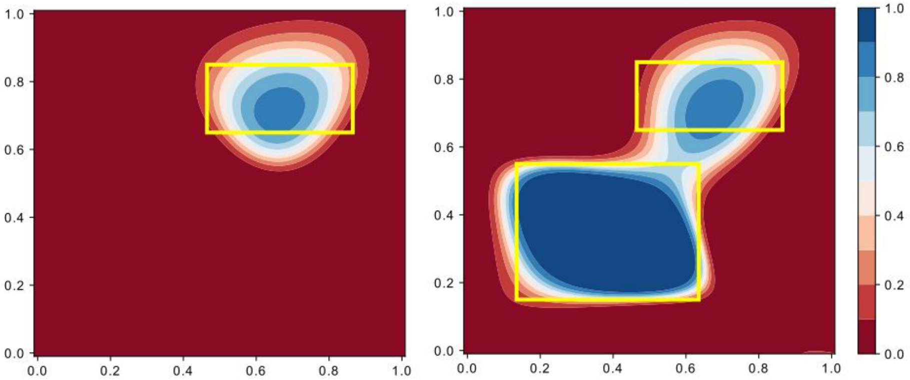
Our Solution: Back to the Example

Two classes: A, B

Constraint: $A \to B$

Standard solution:

Class A Class B



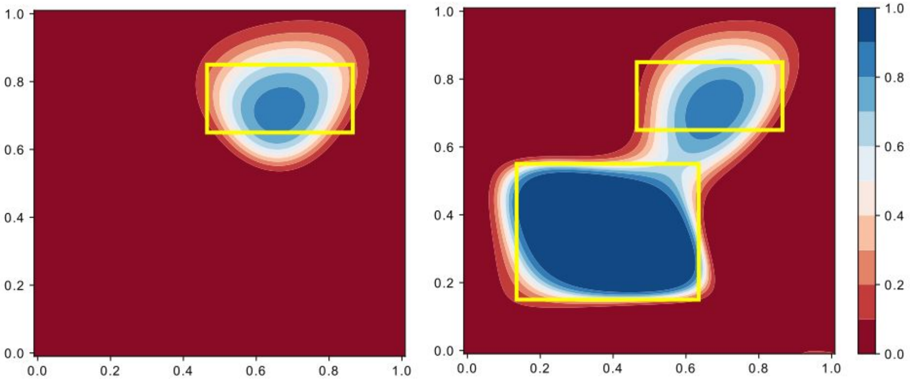
Our Solution: Back to the Example

Two classes: A, B

Constraint: $A \to B$

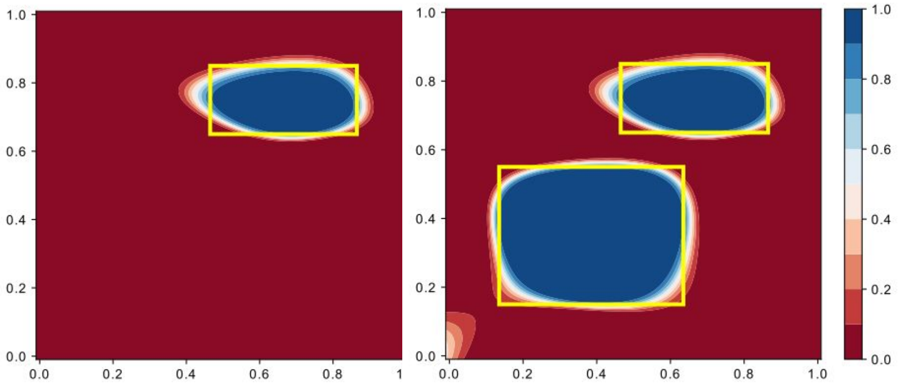
Standard solution:

Class A Class B

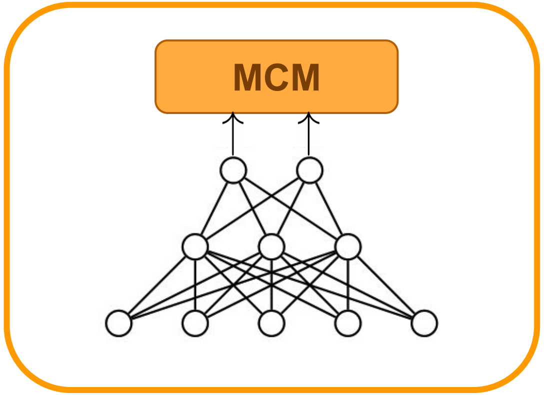


Our solution:

Class A Class B



Our Solution: Back to the Example

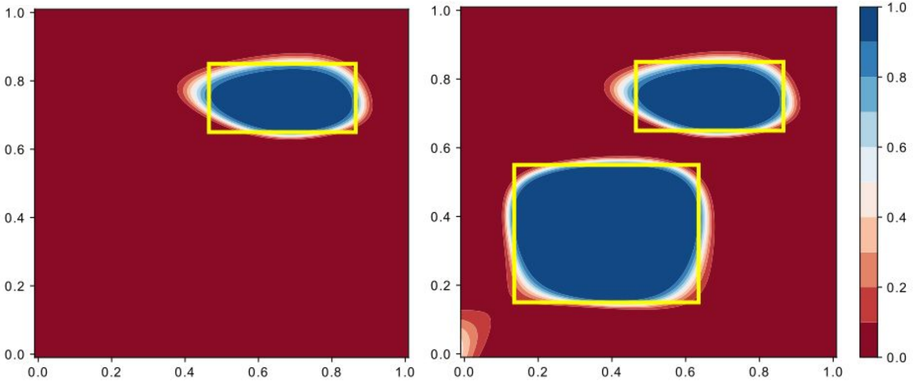


Two classes: A, B

Constraint: $A \rightarrow B$

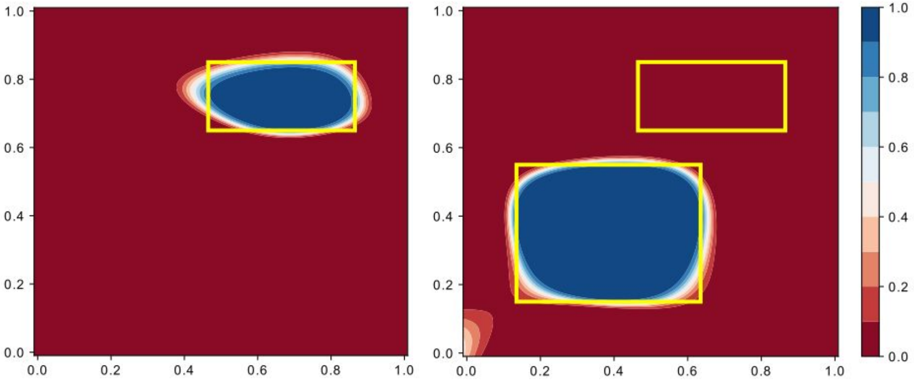
Output After Layer

Class A Class B



Output Before Layer:

Class A Class B



Hierarchical Constraints

Max Constraint Layer

For each label A

$$\text{MCM}_A = \max_{B \in \mathcal{D}_A} (h_B)$$

where $\mathcal{D}_{\mathcal{A}}$ is the set of of subclasses of A

Hierarchical Constraints

Max Constraint Layer

For each label A

$$\mathrm{MCM}_A = \max_{B \in \mathcal{D}_A} (h_B)$$

where $\mathcal{D}_A$ is the set of of subclasses of A

Max Constraint Loss

$$\mathrm{MCLoss} = \sum_{A \in \mathcal{A}} \mathrm{MCLoss}_A$$

$$\mathrm{MCLoss}_A = -y_A \ln(\max_{B \in \mathcal{D}_A} (y_B h_B)) - (1 - y_A) \ln(1 - \mathbf{MCM}_A).$$

Experimental Analysis

Dataset	CCN(h)	HMC-LMLP	Clus-Ens	HMCN-R	HMCN-F
Cellcycle FUN	**0.255**	0.207	0.227	0.247	0.252
Derisi FUN	**0.195**	0.182	0.187	0.189	0.193
Eisen FUN	**0.306**	0.245	0.286	0.298	0.298
Expr FUN	**0.302**	0.242	0.271	0.300	0.301
Gasch1 FUN	**0.286**	0.235	0.267	0.283	0.284
Gasch2 FUN	**0.258**	0.211	0.231	0.249	0.254
Seq FUN	**0.292**	0.236	0.284	0.290	0.291
Spo FUN	**0.215**	0.186	0.211	0.210	0.211
Cellcycle GO	**0.413**	0.361	0.387	0.395	0.400
Derisi GO	**0.370**	0.343	0.361	0.368	0.369
Eisen GO	**0.455**	0.406	0.433	0.435	0.440
Expr GO	0.447	0.373	0.422	0.450	**0.452**
Gasch1 GO	**0.436**	0.380	0.415	0.416	0.428
Gasch2 GO	0.414	0.371	0.395	0.463	**0.465**
Seq GO	0.446	0.370	0.438	0.443	**0.447**
Spo GO	**0.382**	0.342	0.371	0.375	0.376
Diatoms	**0.758**	-	0.501	0.514	0.530
Enron	**0.756**	-	0.696	0.710	0.724
Imclef07a	**0.956**	-	0.803	0.904	0.950
Imclef07d	**0.927**	-	0.881	0.897	0.920
Average Ranking	1.25	5.00	3.93	2.93	1.90

Outline

1. Introduction
2. Hierarchical Constraints [1]
3. Normal Logic Constraints [2]
4. A Novel Benchmark for Neuro-symbolic Models [3]
5. Summary and Open Questions
6. Q&A

[1] Giunchiglia E., Lukasiewicz T., Coherent Hierarchical Multi-label Classification Networks, NeurIPS, 2020
[2] Giunchiglia E., Lukasiewicz T., Multi-label Classification Neural Networks with Hard Logical Constraints, JAIR, 2021
[3] Giunchiglia E., Stoian M., Khan S., Cuzzolin F., Lukasiewicz T., ROAD-R: The Autonomous Driving Dataset with Logical Requirements,MLJ, 2022

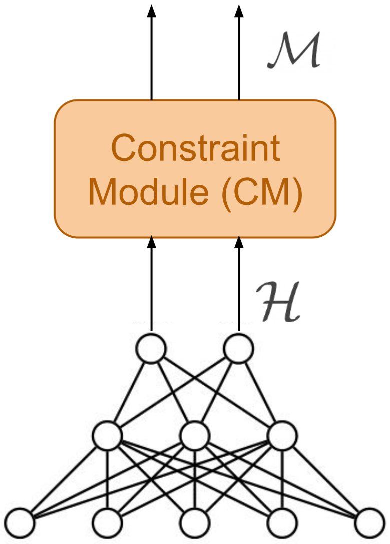
Constraints as Normal Logic Rules

$$A_1, \ldots, A_k, \neg A_{k+1}, \ldots, \neg A_n \rightarrow A$$

If a datapoint is associated with the labels $A_1, \ldots, A_k$ and not with the labels $A_{k+1}, \ldots, A_n$ then it must be associated with the label A

with $1 \leq k < k+1 \leq n$

Goal



Given a set of constraints Π the final output $\mathcal{M}$ should be:

- coherent with Π
- supported relative to $\mathcal{H}$ and Π
- minimal relative to $\mathcal{H}$ and Π
- unique

Can we use the same idea as in the hierarchical case?

Hierarchy Constraints:

$$A \to B$$

$$\mathrm{MCM}_A = h_A$$
$$\mathrm{MCM}_B = \max(h_B, h_A)$$

Can we use the same idea as in the hierarchical case?

Hierarchy Constraints:

$$A \rightarrow B$$

$$\mathrm{MCM}_A = h_A$$
$$\mathrm{MCM}_B = \max(h_B, h_A)$$

Normal Logic Constraints:

$$A, \neg B \rightarrow C$$

$$\mathrm{CM}_A = h_A$$
$$\mathrm{CM}_B = h_B$$
$$\mathrm{CM}_C = \max(h_C, \min(h_A, 1 - h_B))$$

Can we use the same idea as in the hierarchical case?

Hierarchy Constraints:

$$A \to B$$

$$\text{MCM}_A = h_A$$
$$\text{MCM}_B = \max(h_B, h_A)$$

Normal Logic Constraints:

$$A, \neg B \to C$$

$$\text{CM}_A = h_A$$
$$\text{CM}_B = h_B$$
$$\text{CM}_C = \max(h_C, \min(h_A, 1 - h_B))$$

Can this work with multiple constraints?

Can we use the same idea as in the hierarchical case?

Hierarchy Constraints:

$$A \to B$$

$$\mathrm{MCM}_A = h_A$$
$$\mathrm{MCM}_B = \max(h_B, h_A)$$

Normal Logic Constraints:

$$A, \neg B \to C$$

$$\mathrm{CM}_A = h_A$$
$$\mathrm{CM}_B = h_B$$
$$\mathrm{CM}_C = \max(h_C, \min(h_A, 1 - h_B))$$

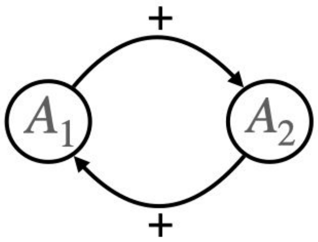
Can this work with multiple constraints?

No!

Problems

Circularities

$$A_1 \to A_2$$
$$A_2 \to A_1$$



$$m_{A_1} = \max\left(h_{A_1}, m_{A_2}\right)$$
$$m_{A_2} = \max\left(h_{A_2}, m_{A_1}\right)$$

Easy to solve: take the minimum of the set of the tuples of values satisfying the equations.
In the example above:

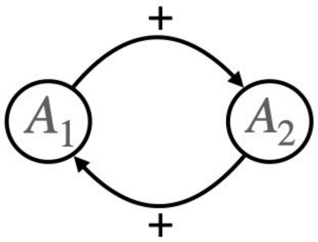
$$m_{A_1} = m_{A_2} = \max\left(h_{A_1}, h_{A_2}\right)$$

Problems

Circularities

$A_1 \to A_2$

$A_2 \to A_1$



$$m_{A_1} = \max(h_{A_1}, m_{A_2})$$
$$m_{A_2} = \max(h_{A_2}, m_{A_1})$$

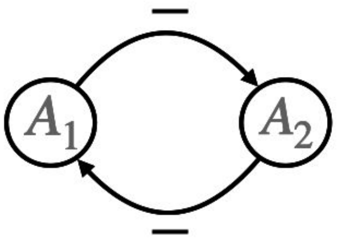
Easy to solve: **take the minimum of the set of the tuples of values satisfying the equations.**
In the example above:

$$m_{A_1} = m_{A_2} = \max(h_{A_1}, h_{A_2})$$

Negations

$\neg A_1 \to A_2$

$\neg A_2 \to A_1$

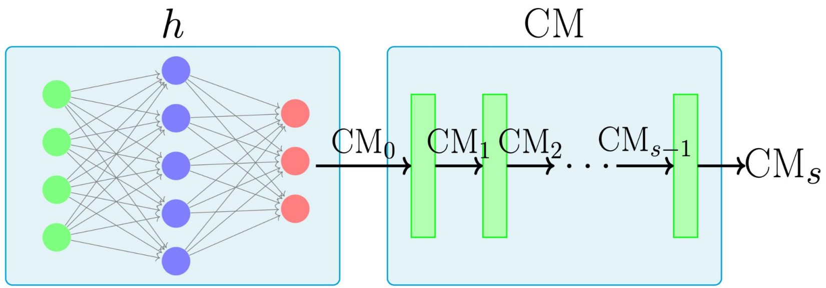


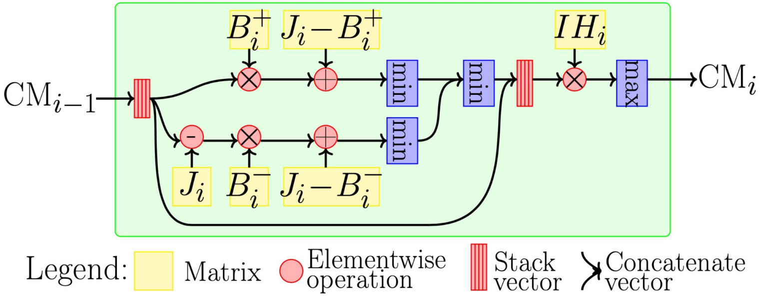
$$m_{A_1} = \max(h_{A_1}, 1 - m_{A_2})$$
$$m_{A_2} = \max(h_{A_2}, 1 - m_{A_1})$$

Additional assumption needed: **the set of constraints Π needs to be *stratified*.**

The set of constraints $\{\neg A_1 \to A_2, \neg A_2 \to A_1\}$ is not stratified.

Constraint Layer





$\neg A \rightarrow B$

$\neg B \rightarrow C$

$\Pi_1 = \emptyset$

$\Pi_2 = \{\neg A \rightarrow B\}$

$\Pi_3 = \{\neg B \rightarrow C\}$

1st stratum: $\mathrm{CM}_A = h_A$

2nd stratum: $\mathrm{CM}_B = \max(h_B, 1 - \mathrm{CM}_A)$

3rd stratum: $\mathrm{CM}_C = \max(h_C, 1 - \mathrm{CM}_B)$

Experimental Analysis

Model	ARTS	BUSINESS	CAL500	EMOTIONS	ENRON	GENBASE	IMAGE	MEDICAL
				Average precision (↑)				
CCN(h)	0.623	**0.904**	**0.520**	**0.800**	0.704	0.996	**0.807**	**0.866**
CAMEL	**0.625**	0.899	0.513	0.756	**0.708**	0.990	0.793	0.807
ECC	0.544	0.867	0.401	0.772	0.643	**1.000**	0.738	0.823
BR	0.546	0.863	0.441	0.793	0.643	**1.000**	0.726	0.823
RAKEL	0.530	0.856	0.433	0.798	0.636	**1.000**	0.721	0.811
				Coverage error (↓)				
CCN(h)	**0.172**	**0.065**	**0.734**	**0.315**	**0.217**	0.016	**0.187**	**0.035**
CAMEL	0.202	0.083	0.791	0.372	0.256	0.010	0.201	0.036
ECC	0.223	0.089	0.853	0.338	0.285	**0.009**	0.242	0.045
BR	0.217	0.086	0.789	0.324	0.288	**0.009**	0.245	0.045
RAKEL	0.221	0.085	0.791	0.317	0.294	**0.009**	0.250	0.049
				Hamming loss (↓)				
CCN(h)	**0.054**	**0.023**	**0.136**	**0.197**	**0.046**	**0.001**	**0.172**	**0.013**
CAMEL	0.055	**0.023**	0.138	0.265	0.047	0.003	0.174	0.024
ECC	0.081	0.031	0.172	0.245	0.055	**0.001**	0.218	0.019
BR	0.079	0.032	0.162	0.229	0.054	**0.001**	0.232	0.019
RAKEL	0.082	0.034	0.165	0.223	0.055	**0.001**	0.225	0.019
				Multi-label accuracy (↑)				
CCN(h)	**0.238**	0.601	0.203	**0.534**	**0.395**	0.986	**0.488**	**0.589**
CAMEL	0.218	**0.609**	0.210	0.354	0.381	0.943	0.456	0.284
ECC	0.217	0.548	0.220	0.446	0.361	**0.992**	0.387	0.481
BR	0.217	0.538	0.221	0.465	0.365	**0.992**	0.369	0.477
RAKEL	0.215	0.527	**0.222**	0.485	0.361	**0.992**	0.376	0.481
				One-error (↓)				
CCN(h)	0.475	0.093	**0.113**	**0.273**	0.235	**0.000**	**0.296**	**0.181**
CAMEL	**0.460**	**0.090**	0.133	0.381	**0.223**	0.020	0.310	0.285
ECC	0.568	0.137	0.378	0.332	0.309	**0.000**	0.392	0.251
BR	0.567	0.147	0.232	0.292	0.299	**0.000**	0.425	0.251
RAKEL	0.586	0.159	0.232	0.292	0.304	**0.000**	0.430	0.266
				Ranking loss (↓)				
CCN(h)	**0.115**	**0.030**	**0.173**	**0.161**	**0.076**	0.003	**0.159**	**0.024**
CAMEL	0.136	0.040	0.189	0.237	0.086	**0.001**	0.177	0.026
ECC	0.158	0.046	0.257	0.193	0.107	**0.001**	0.231	0.033
BR	0.155	0.045	0.218	0.177	0.108	**0.001**	0.234	0.032
RAKEL	0.159	0.044	0.220	0.169	0.112	**0.001**	0.242	0.037

Model	RCV1S1	RCV1S2	RCV1S3	RCV1S4	RCV1S5	SCIENCE	SCENE	YEAST
				Average precision (↑)				
CCN(h)	**0.642**	**0.666**	**0.647**	**0.675**	0.560	0.603	**0.868**	**0.768**
CAMEL	0.622	0.647	0.636	0.654	**0.564**	**0.614**	0.824	0.766
ECC	0.549	0.575	0.585	0.609	0.529	0.502	0.794	0.724
BR	0.536	0.563	0.572	0.600	0.524	0.500	0.781	0.743
RAKEL	0.532	0.556	0.562	0.589	0.508	0.493	0.794	0.732
				Coverage error (↓)				
CCN(h)	**0.092**	**0.089**	**0.103**	**0.080**	**0.107**	**0.131**	**0.077**	**0.452**
CAMEL	0.131	0.115	0.123	0.103	0.130	0.162	0.106	0.457
ECC	0.185	0.166	0.167	0.169	0.196	0.225	0.127	0.495
BR	0.194	0.181	0.178	0.184	0.210	0.227	0.128	0.476
RAKEL	0.201	0.180	0.185	0.195	0.209	0.225	0.123	0.481
				Hamming loss (↓)				
CCN(h)	**0.026**	**0.022**	**0.024**	**0.019**	**0.025**	**0.031**	**0.092**	**0.196**
CAMEL	0.027	**0.022**	**0.024**	0.021	**0.025**	**0.031**	0.109	**0.196**
ECC	0.031	0.027	0.028	0.026	0.030	0.049	0.131	0.221
BR	0.032	0.028	0.029	0.027	0.031	0.051	0.151	0.214
RAKEL	0.033	0.029	0.030	0.027	0.031	0.051	0.130	0.225
				Multi-label accuracy (↑)				
CCN(h)	**0.296**	**0.310**	**0.303**	**0.324**	**0.275**	**0.255**	**0.607**	**0.480**
CAMEL	0.204	0.222	0.210	0.257	0.223	0.217	0.528	**0.480**
ECC	0.264	0.277	0.273	0.297	0.269	0.209	0.478	0.443
BR	0.263	0.279	0.275	0.289	0.263	0.200	0.438	0.456
RAKEL	0.263	0.272	0.268	0.290	0.258	0.201	0.481	0.445
				One-error (↓)				
CCN(h)	**0.413**	**0.389**	**0.405**	**0.379**	**0.402**	0.494	**0.224**	0.234
CAMEL	**0.413**	0.397	0.413	0.399	0.414	**0.472**	0.287	**0.231**
ECC	0.477	0.462	0.453	0.436	0.451	0.603	0.319	0.300
BR	0.492	0.474	0.466	0.435	0.466	0.605	0.358	0.259
RAKEL	0.488	0.481	0.471	0.439	0.474	0.606	0.329	0.270
				Ranking loss (↓)				
CCN(h)	**0.036**	**0.035**	**0.046**	**0.035**	**0.046**	**0.094**	**0.073**	**0.172**
CAMEL	0.051	0.048	0.050	0.046	0.054	0.117	0.101	0.173
ECC	0.086	0.078	0.078	0.086	0.093	0.176	0.103	0.208
BR	0.091	0.088	0.085	0.097	0.101	0.177	0.131	0.190
RAKEL	0.093	0.088	0.089	0.103	0.102	0.180	0.127	0.200

Outline

1. Introduction
2. Hierarchical Constraints [1]
3. Normal Logic Constraints [2]
4. A Novel Benchmark for Neuro-symbolic Models [3]
5. Open Questions
6. Q&A

[1] Giunchiglia E., Lukasiewicz T., Coherent Hierarchical Multi-label Classification Networks, NeurIPS, 2020
[2] Giunchiglia E., Lukasiewicz T., Multi-label Classification Neural Networks with Hard Logical Constraints, JAIR, 2021
[3] Giunchiglia E., Stoian M., Khan S., Cuzzolin F., Lukasiewicz T., ROAD-R: The Autonomous Driving Dataset with Logical Requirements,MLJ, 2022

A Novel Benchmark for Neuro-symbolic Models

Problem: no realistic safety-critical dataset was annotated with logical constraints!

How can we test novel neuro-symbolic models?

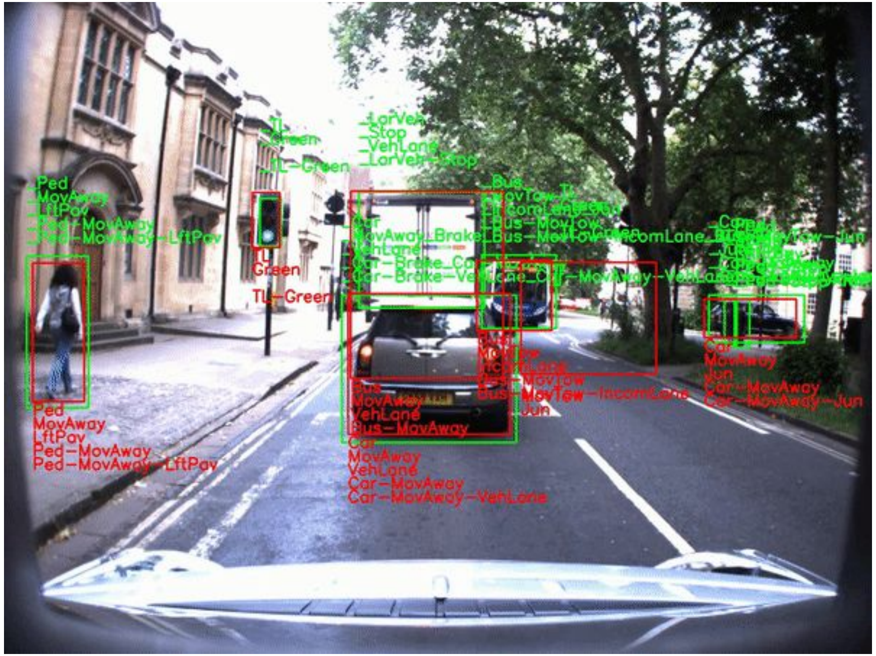
A Novel Benchmark for Neuro-symbolic Models

Problem: no realistic safety-critical dataset was annotated with logical constraints!

How can we test novel neuro-symbolic models?

Solution: we created the first autonomous driving dataset with requirements expressed as logical constraints

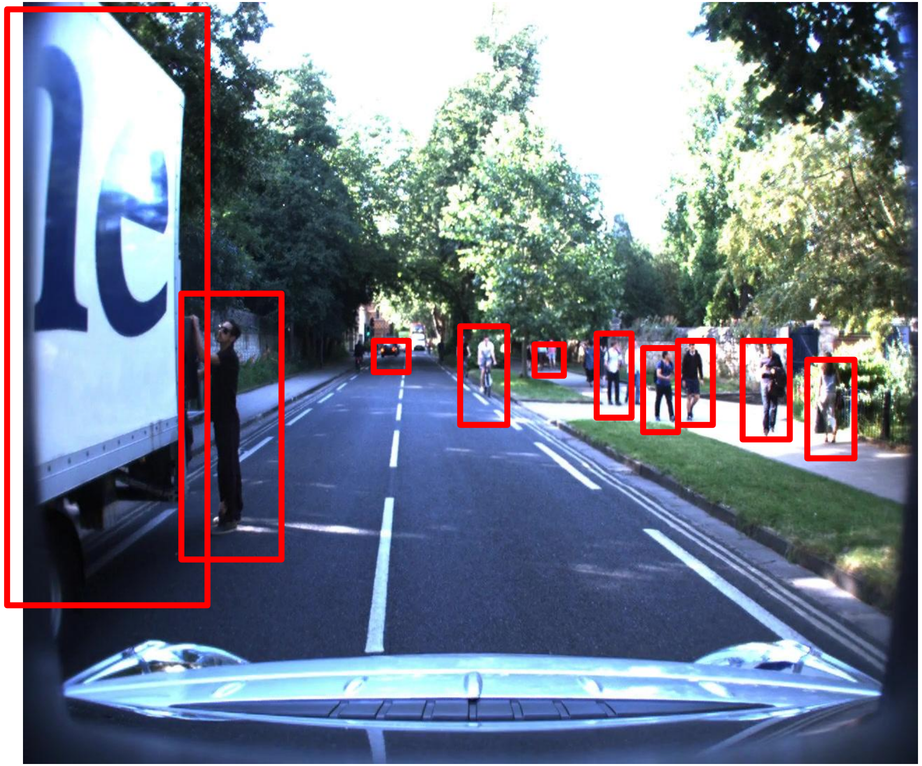
Application Domain: Self-driving Cars



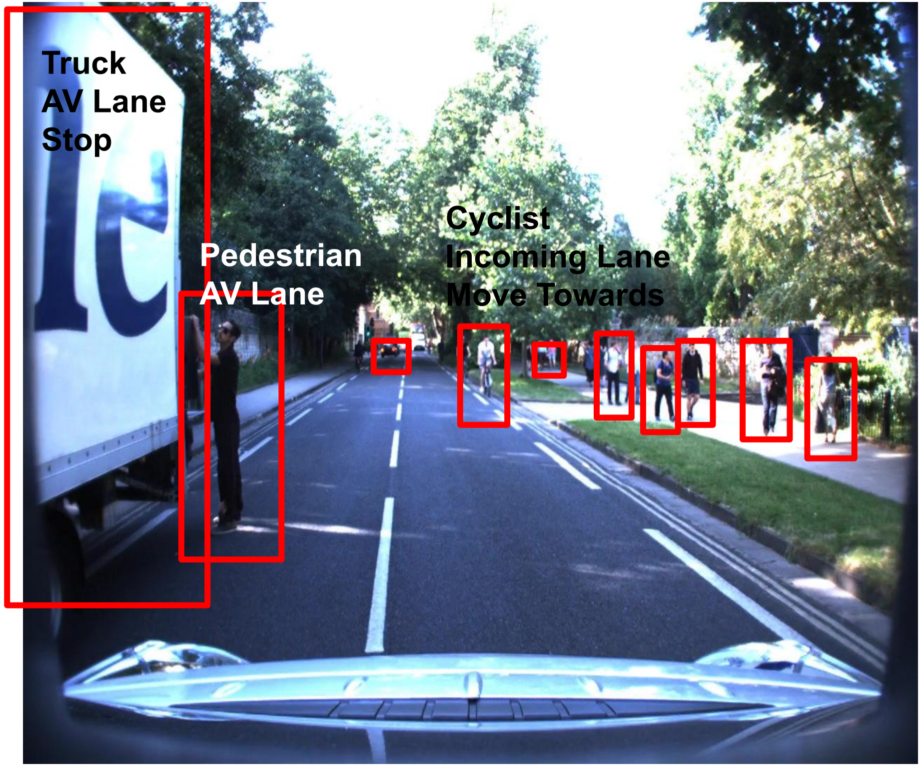
ROAD [5]: multi-label classification dataset for autonomous driving

[5] Gurkirt Singh et al. ROAD: The road event awareness dataset for autonomous driving. TPAMI, 2022

Dataset Annotations



Dataset Annotations



ROAD-R: Our Annotations

We annotated ROAD with 243 logical requirements on the output space expressed as propositional logic rules

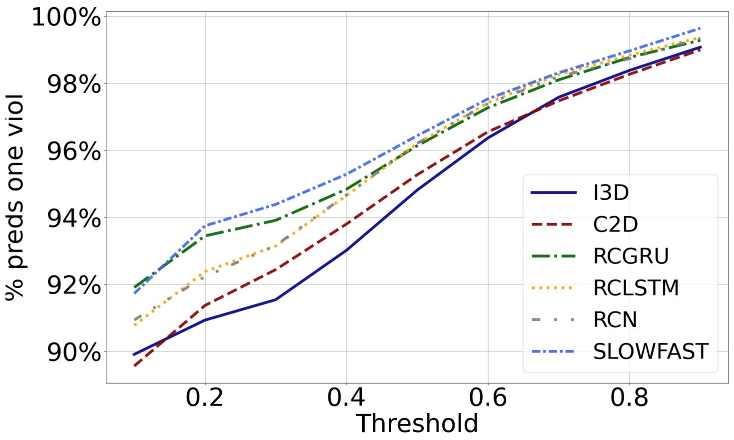
The requirements define the space of the admissible outputs

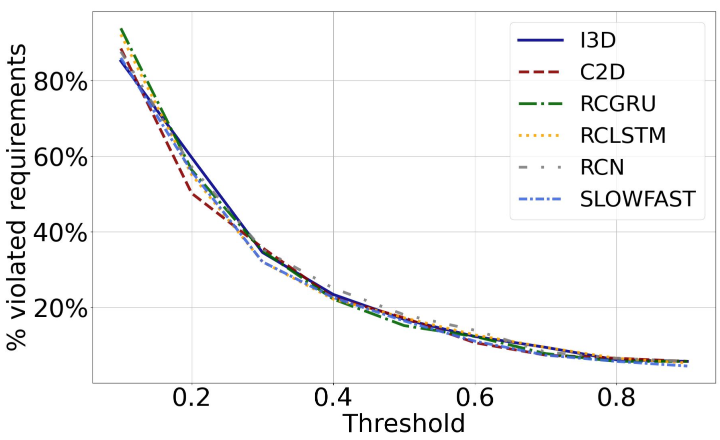
Some examples of constraints:

- *A traffic light cannot be red and green at the same time*
- *An agent cannot move away and towards you at the same time*
- *An agent is either on the right pavement or on the left pavement*

Quantitative Results

Neural networks very often violate even such simple requirements





Qualitative Results

Available Baselines

- Incorporate the constraints during the training phase in the loss

  Pros: neural network can learn from the constraints

  Cons: no guarantee that the constraints will be satisfied

- Include the post-processing at inference time

  Pros: the constraints are guaranteed to be satisfied

  Cons: the neural network is “unaware” of the post-processing step

Neuro-symbolic Baseline vs Standard NN

Comparison of the performance in terms of *frame mean average precision* (f-mAP) between the standard models and the same models trained with the requirements loss and with post-processing

Model	Standard Models	Neuro-symbolic Baseline
C2D	27.57	**28.16**
I3D	30.12	**31.21**
RCGRU	30.78	**31.81**
RCLSTM	30.49	**31.65**
RCN	29.64	**31.02**
SlowFast	28.79	**28.98**

Outline

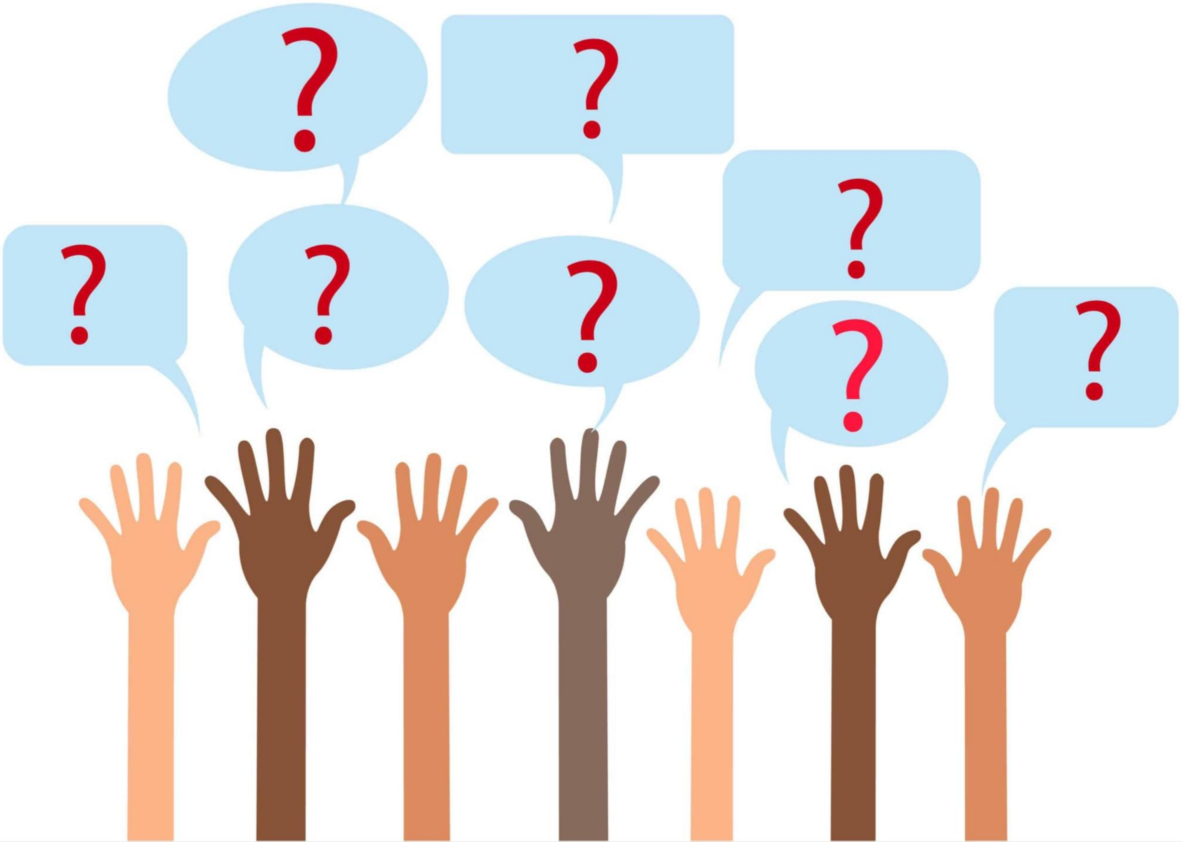
1. Introduction
2. Hierarchical Constraints [1]
3. Normal Logic Constraints [2]
4. Case study: Self-Driving Cars [3]
5. Open Questions
6. Q&A

[1] Giunchiglia E., Lukasiewicz T., Coherent Hierarchical Multi-label Classification Networks, NeurIPS, 2020
[2] Giunchiglia E., Lukasiewicz T., Multi-label Classification Neural Networks with Hard Logical Constraints, JAIR, 2021
[3] Giunchiglia E., Stoian M., Khan S., Cuzzolin F., Lukasiewicz T., ROAD-R: The Autonomous Driving Dataset with Logical Requirements,MLJ, 2022

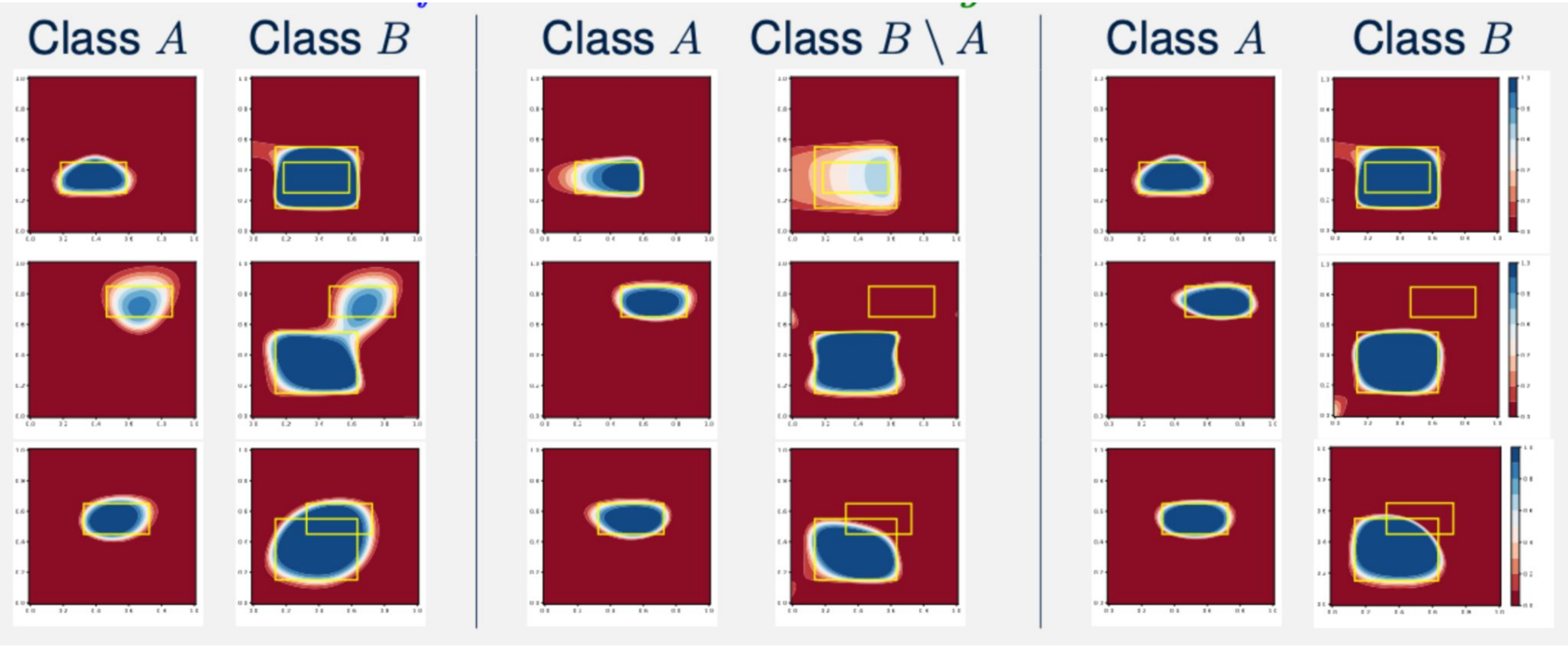
Open Questions

- How can we integrate even more expressive constraints?
- Is it possible to seemingly integrate hard and soft constraints?
- Can we integrate not only constraints over the output space, but also input/output constraints?
- Can we extend this approach to other problem types (e.g., binary classification, regression etc.)?

Thank you! Questions?



Other synthetic experiments



Dependency graph

The dependency graph G_Π of Π is the directed graph having the set of labels as nodes and with, for each constraint $r \in \Pi$,

1. a positive edge from each class in $body^+(r)$ to $head(r)$,
2. a negative edge from each class A such that $\neg A \in body^-(r)$ to $head(r)$.

Stratification

A set of constraints Π is stratified if there is a partition $\Pi_1, \ldots, \Pi_s$ of Π, with Π_1 possibly empty, such that, for every $i \in \{1, \ldots, s\}$

- **For every $A \in \bigcup_{r \in \Pi_i} body^+(r)$, all the constraints with head A in Π belong to $\cup_{j=1}^{i} \Pi_i$**
- **For every $A \in \bigcup_{r \in \Pi_i} body^-(r)$, all the constraints with head A in Π belong to $\cup_{j=1}^{i-1} \Pi_i$**

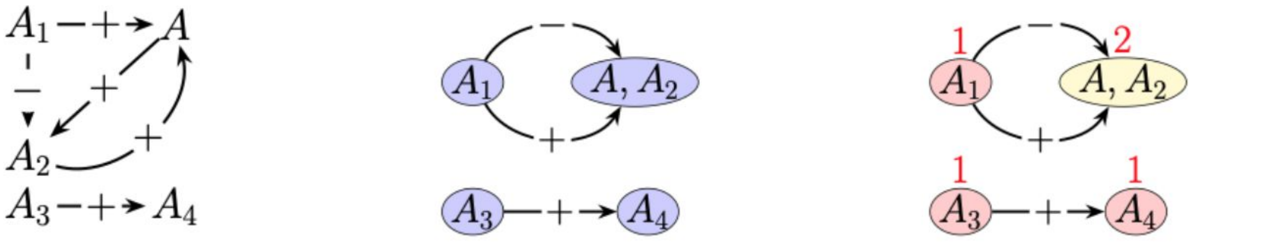
$\Pi_1, \ldots, \Pi_s$ **is a *stratification* of Π, and each Π_i is a *stratum.***

Supported Set of Classes

Let $(\mathcal{P}, \Pi)$ be an LCMC problem. Let h be a model for $\mathcal{P}$. Let $\mathcal{H}$ be the set of classes predicted by h. A set of classes $\mathcal{M}$ is supported relative to $\mathcal{H}$ and Π if for any class $A \in \mathcal{M}$, $A \in \mathcal{H}$, or there exists a constraint $r \in \Pi$ such that $head(r) = A$, $body^+(r) \subseteq \mathcal{M}$, and for each $B \in body^-(r)$, $B \notin \mathcal{M}$.

Dependency graph

$$\Pi = \{A_1 \to A;\ A_2 \to A;\ A, \neg A_1 \to A_2;\ A_3 \to A_4\}$$



(a) G_Π (b) DAGs from step 1 (c) Numbers from step 2

$$\mathcal{A}_1 = \{A_1, A_3, A_4\},\ \mathcal{A}_2 = \{A, A_2\}$$

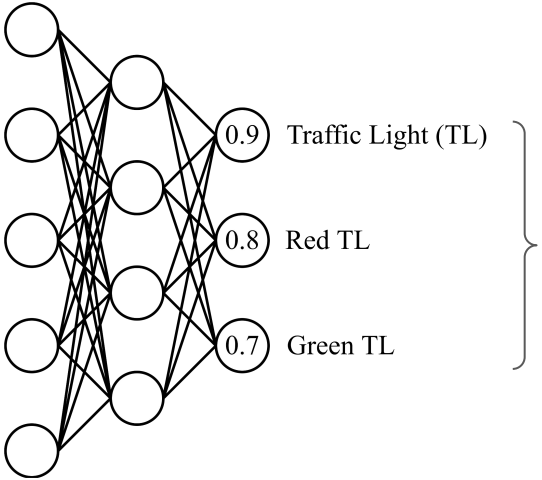
$$\Pi_1 = \{A_3 \to A_4\} \qquad \Pi_2 = \Pi \setminus \Pi_1$$

T-norms

Operation	Minimum/Godel	Product	Lukasiewicz
$\neg x$	$1 - x$	$1 - x$	$1 - x$
$x \wedge y$	$\min(x, y)$	$x \cdot y$	$\max(0, x + y - 1)$
$x \vee y$	$\max(x, y)$	$x + y - x \cdot y$	$\min(1, x + y)$
$x \Rightarrow y$	$x < y ? 1 : y$	$\min(1, \frac{y}{x})$	$x < y ? 1 : 1 - (x - y)$

Inference time: Post-processing step

The problem of finding the *optimal correction* of a prediction can be formulated as a weighted partial maximum satisfiability (PMaxSat) problem. **How?**



Prediction:

$\{\mathrm{TL}, \mathrm{GreenTL}, \mathrm{RedTL}\}$

$$\mathrm{GreenTL} \rightarrow \mathrm{TL}$$
$$\mathrm{RedTL} \rightarrow \mathrm{TL}$$
$$\mathrm{GreenTL} \rightarrow \neg\mathrm{RedTL}$$

$$\mathrm{cost} = \sum_i c_i$$

PMaxSat formulation:

Hard clauses:

$\neg\mathrm{GreenTL} \vee \mathrm{TL}$

$\neg\mathrm{RedTL} \vee \mathrm{TL}$

$\neg\mathrm{GreenTL} \vee \neg\mathrm{RedTL}$

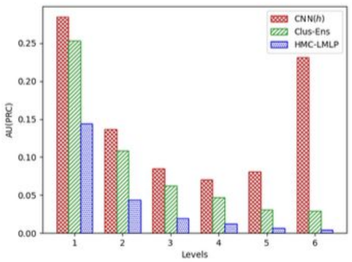
Soft Clauses:

$c_1 \mathrm{TL}$

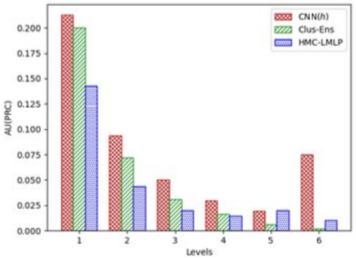
$c_2\ \mathrm{RedTL}$

$c_3 \mathrm{GreenTL}$

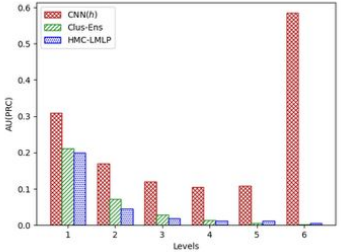
CCN Performance by Level



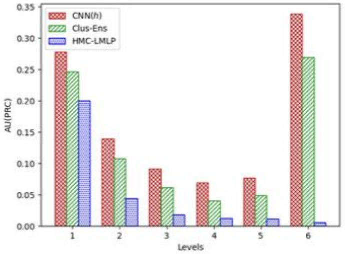
(a) CELLCYCLE FUN



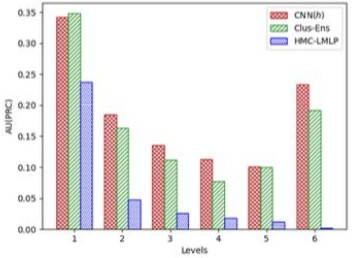
(b) DERISI FUN



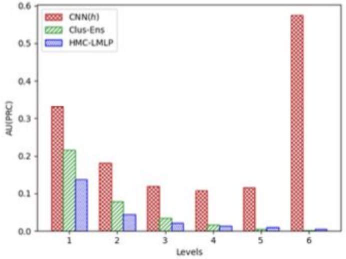
(e) GASCH1 FUN



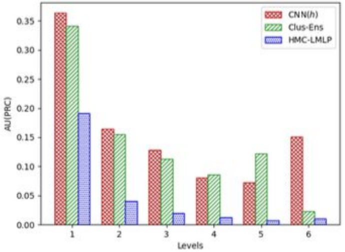
(f) GASCH2 FUN



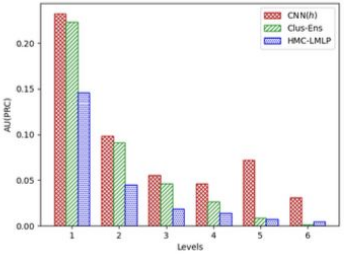
(c) EISEN FUN



(d) EXPR FUN



(g) SEQ FUN



(h) SPO FUN

How can we avoid “meaningless” mistakes?

Requirements specification is a key step in standard software development

→ Machine learning models development requires the same step