ARTICLE IN PRESS

International Journal of Approximate Reasoning xxx (xxxx) xxx

Contents lists available at ScienceDirect

International Journal of Approximate Reasoning

journal homepage: www.elsevier.com/locate/ijar

CCN$^+$: A neuro-symbolic framework for deep learning with requirements

Eleonora Giunchiglia [a,*], Alex Tatomir [b], Mihaela Cătălina Stoian [b], Thomas Lukasiewicz [a,b]

[a] Institute of Logic and Computation, TU Wien, Vienna, Austria
[b] Department of Computer Science, University of Oxford, Oxford, UK

ARTICLE INFO

Dataset link: https://github.com/atatomir/CCN

Keywords:
Machine learning with requirements
Neuro-symbolic AI
Machine learning
Deep learning
Safe AI
Artificial intelligence

ABSTRACT

For their outstanding ability of finding hidden patterns in data, deep learning models have been extensively applied in many different domains. However, recent works have shown that, if a set of requirements expressing inherent knowledge about the problem at hand is given, then neural networks often fail to comply with them. This represents a major drawback for deep learning models, as requirements compliance is normally considered a necessary condition for standard software deployment. In this paper, we propose a novel neuro-symbolic framework able to make any neural network compliant by design to a given set of requirements over the output space expressed in full propositional logic. This framework, called CCN$^+$, integrates the requirements into the output layer of the neural network by applying multiple inference rules that ensure compliance with the requirements and adapts the standard binary cross-entropy loss function to the requirement output layer. As a result, not only the outputted predictions are guaranteed to be compliant with the requirements, but the neural network itself learns how to exploit the domain knowledge expressed by the requirements to get better performance. We conduct an extensive experimental evaluation of CCN$^+$ on 19 real-world multi-label classification datasets with propositional logic requirements, including a challenging dataset for autonomous driving. Our experimental analysis confirms that CCN$^+$ is able to outperform both its neural counterparts and the state-of-the-art models.

1. Introduction

Neural networks are exceptional tools for finding correlations in raw data and turning them into accurate predictions. Thanks to this ability, they have been applied in a vast range of different domains, from facial recognition [1] and financial forecasting [2] to product recommendation [3] and medical sciences [4]. However, in many application domains the deployed models have to satisfy (or even be certified to satisfy) a given set of requirements about the problem at hand, and recent works [5–7] have actually shown that neural networks often fail to comply with them.

To address the above shortcoming, requirements were initially treated as logical formulas and compiled into the loss function of the neural network at hand (see, e.g., [8–10]). These loss-based approaches were able to reduce the number of violations of the requirements, but they did not give any guarantee that the obtained models would indeed satisfy them. Thus, multiple works

* Corresponding author.
E-mail address: eleonora.giunchiglia@tuwien.ac.at (E. Giunchiglia).

https://doi.org/10.1016/j.ijar.2024.109124
Received 12 March 2023; Received in revised form 10 January 2024; Accepted 11 January 2024

Available online 22 January 2024

(see, e.g., [11–13]) in hierarchical multi-label classification (HMC) and some works in the neuro-symbolic field [5,6,14] proposed to incorporate the requirements as constraints in the topology of the network itself, which allowed to guarantee their satisfaction. However, all such approaches either can only consider constraints with limited expressivity (see, e.g., [11–13,5]), or are built in such a way that each output has to correspond to one admissible configuration of the system outputs and indeed there can be exponentially many such configurations (see, e.g., [6]), or are not able to obtain the same performance as our model as demonstrated by our experimental analysis (see, e.g., [14]).

In this paper, we extend [13,5] and propose a novel neuro-symbolic framework able to make any neural network for multi-label classification compliant by design to a given set of requirements expressible in propositional logic. Similarly to [13,5], this framework, called Coherent-by-Construction Network$^+$ (CCN$^+$), integrates the requirements as rules into the output layer of the neural network, and adapts the standard binary cross-entropy loss function to the requirement output layer. As a result, not only the outputted predictions are guaranteed to be compliant with the requirements, but the neural network itself learns how to exploit the domain knowledge expressed by the requirements to get better performance. Differently from [13,5], in CCN$^+$, requirements are not expressed as rules, but as arbitrary clauses in propositional logic, and then converted into a corresponding set of rules with a possibly negative head. Thus, CCN$^+$ does not suffer from the limitations in the expressivity affecting the models presented in [13,5]. At the same time, the conversion from clauses to a corresponding set of rules and the presence of rules with positive/negative heads poses both theoretical questions (like how to avoid firing two rules with contradictory heads) and practical issues (like how to convert clauses into rules while maximizing performances). In order to highlight the differences in the formal treatment of rules with positive vs. negative heads, and also as a motivating example showing the difference it may make how constraints are mapped into rules, we initially consider a single binary constraint $(\neg A_1 \vee A_2)$ together with two synthetic datasets and discuss the pros and cons of converting it into the rule $(A_1 \rightarrow A_2)$ (enabling the computation of A_2 on the basis of A_1) or into the rule $(\neg A_2 \rightarrow \neg A_1)$ (enabling the computation of $\neg A_1$ on the basis of $\neg A_2$). Then, we consider an arbitrary set of requirements, generalize the proposed solution, and show how it is possible to efficiently handle a high number of requirements by parallelizing the computation of the labels with GPUs.

To test the model in a real-world and safety-critical scenario, we deploy it on the ROAD-R dataset [15], which is an extension of the ROAD dataset [16]. ROAD consists of 22 videos with 122 000 frames recorded from the perspective of a vehicle, labeling multiple agents with their locations and actions. ROAD was subsequently extended to form ROAD-R in [15], as it was further annotated with 243 requirements expressing facts such as “a traffic light can not be red and green at the same time”. Surprisingly, as shown in [15], even though the initial unconstrained model achieves high performance, more than about 90% of its predictions violate at least one requirement, regardless of the threshold used. We then consider different procedures for converting the constraints into rules and show that not only the resulting model never commits a violation, but it also outperforms its unconstrained counterpart by up to 7.2%.

Additionally, we then test the abilities of CCN$^+$ on the 18 real world multi-labels datasets proposed in [5]. Such datasets represent a comprehensive test-bed for our model as they belong to different application domains and differ significantly both in the number of data points/classes and in the characteristics of the associated sets of constraints. Further, contrarily to ROAD-R, they come with requirements expressed as stratified normal rules which allows us to compare CCN$^+$ with CCN(h), i.e., our previous model proposed in [5], showing that it is able to keep its top performance while being able to handle more expressive requirements. In addition to CCN(h), we compare CCN$^+$ with Semantic Probabilistic Layer (SPL), i.e., the current state-of-the-art model for the enforcement of propositional logic requirements, and 4 state-of-the-art models for multi-label classification problems whose predictions are made compliant with the requirements via post-processing, and we are able to show that CCN$^+$ outperforms all of them according to 6 different metrics.

The main contributions of this paper can be thus briefly summarized as follows.

- We propose a novel neuro-symbolic framework, called CCN$^+$, which extends the works in [13,5] by allowing to handle arbitrary requirements expressed in propositional logic. In CCN$^+$, requirements are converted into rules and CCN$^+$ is able to predict the head of each rule on the basis of the predictions of the labels in the body.
- On the theoretical side, regardless of how requirements are converted into rules, we show that CCN$^+$ (i) is guaranteed to be compliant with the set of requirements; (ii) is able to exploit the requirements also during the training; and (iii) can be easily run on standard GPUs.
- On the experimental side, we consider various procedures for converting requirements into rules, and we perform extensive analysis on both synthetic and real-world datasets, showing the effectiveness of our model. The analysis on ROAD-R shows that our proposal can handle very large problems.

The rest of the paper is structured as follows. First, we start with the necessary preliminaries and notations in Section 2, followed by the presentation of CCN$^+$ in Section 3, and how CCN$^+$ can be optimized for GPUs in Section 4. The experimental analysis is in Section 5, while the related work is discussed in Section 6. We finally wrap up the paper with the concluding remarks and future work in Section 7.

2. Preliminaries and notation

A *multi-label classification (MC) problem* $\mathcal{P}$ is a pair $(\mathcal{A}, \mathcal{X})$, where $\mathcal{A}$ is a finite set of labels, $\mathcal{X}$ is a finite set of pairs $(x, \mathcal{Y})$, with $x \in \mathbb{R}^D (D \geq 1)$ being a data point, and $\mathcal{Y} \subseteq \mathcal{A}$ the ground truth of x, i.e., the set of labels associated with x. A *model* m for $\mathcal{P}$ is a

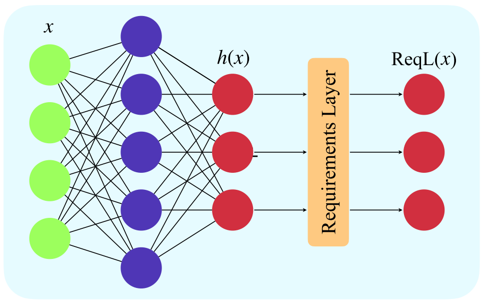


Fig. 1. Basic intuition behind CCN$^+$.

function $m(\cdot,\cdot)$ mapping every label in $\mathcal{A}$ and data point $x \in \mathbb{R}^D$ to $[0,1]$. For every label A, the function $m_A : \mathbb{R}^D \to [0,1]$ is defined by $x \mapsto m(A,x)$, for every data point $x \in \mathbb{R}^D$. Given a data point x, for each label $A \in \mathcal{A}$, we say that A is *predicted* if $m_A(x) > \theta$, θ being a user-defined threshold, while we call *prediction* the set of predicted labels and *output* the vector in $[0,1]^{|\mathcal{A}|}$ returned by the model. For technical reasons, detailed in the next section, we assume that each model m maps every label A and data point x to a value that is either smaller or greater than the threshold θ, i.e., $m(A,x) \neq \theta$.[1] For ease of presentation, in the rest of the paper, we often omit the dependency from data points, and simply write, for example, m_A instead of $m_A(x)$.

Given an MC problem $\mathcal{P}$, we define the acceptable behaviour of a model for $\mathcal{P}$ with respect to a set of requirements expressed as propositional logic formulas over the set $\mathcal{A}$ of labels. As defined in [15], an *MC problem with propositional logic requirements* $(\mathcal{P},\Pi)$ consists of an MC problem $\mathcal{P}$ and a finite, consistent set Π of *requirements* expressed as propositional formulas over the set $\mathcal{A}$ of labels in $\mathcal{P}$. Indeed, we consider only problems in which the set of Π requirements is consistent, otherwise, the problem of guaranteeing compliance with Π would be vacuous. Given an MC problem with requirements $(\mathcal{P},\Pi)$, a model m for $\mathcal{P}$ and a data point x, we say that the prediction made by m for x *satisfies*

1. a simple atomic requirement $A \in \mathcal{A}$ (stating that the label A has always to be predicted) when A is predicted by m;
2. an arbitrary requirement according to the standard truth tables of propositional logic;
3. the set Π of requirements if the prediction made by m satisfies all the requirements in Π.

As an alternative, simpler terminology, we say that m *satisfies* or *predicts a requirement (to be true)* if the prediction made by m for the given data point satisfies it. Finally, as in [5], we say that a model m for $\mathcal{P}$ is *coherent* with Π if all its predictions satisfy all the requirements in Π.

Without loss of generality, we assume that the requirements are given as a set of *clauses*, each of the form

$$l_1 \vee l_2 \vee \cdots \vee l_n, \tag{1}$$

where every l_i is a *literal*, i.e., is either a label $A \in \mathcal{A}$ or the negation $\neg A$ of a label $A \in \mathcal{A}$ ($i \in \{1,\dots,n\}$). Intuitively, (1) expresses the requirement that each model has always to predict at least one of the literals in the clause, i.e., in $\{l_1,\dots,l_n\}$. We assume that in any clause, a label occurs either positively or negatively at most once. A label A *occurs positively* (resp., *negatively*) in the clause (1) if for a literal l in the clause, $l = A$ (resp., $l = \neg A$), and A *occurs* in (1) if A occurs either positively or negatively in (1).

3. Coherent-by-Construction Network$^+$ (CCN$^+$)

This section defines all the modules of the *Coherent-by-Construction Neural Network*$^+$ (CCN$^+$) framework. Given an MC problem with requirements $(\mathcal{P},\Pi)$ and an arbitrary neural network h for the MC problem $\mathcal{P}$, CCN$^+$ is built by extending h with two innovative components:

(i) a *requirements layer* (ReqL) built on top of h that enforces the coherence of the model predictions with the set Π of requirements; and
(ii) a *requirements loss* (ReqLoss) that exploits the constrained structure of the requirements layer to obtain a better performance.

Fig. 1 shows the basic intuition behind the topology of CCN$^+$. Thanks to the requirements layer, for each input x, the neural network generates an initial output $h(x)$ (which is possibly incoherent with Π), and then the requirements layer returns a new output ReqL(x) such that the associated prediction is guaranteed to satisfy Π. The new output is computed by the requirements layer by exploiting the rules corresponding to Π. CCN$^+$ is trained using the requirements loss, which teaches h how to exploit the requirements layer to get better performance.

[1] Practically, this amounts to either adding or subtracting an arbitrarily small quantity to the outputs originally equal to θ.

To get an intuition on how CCN$^+$ is able to exploit the requirements, suppose that we have the simple requirement $A_1 \vee A_2$. This requirement can be read either as a clause telling us that at least one label between A_1 and A_2 must be predicted positively, or it can be read more operationally as: if A_1 (resp., A_2) is not predicted, then A_2 (resp., A_1) must be predicted. Such operational readings can be captured by using the rules:

$$\neg A_1 \rightarrow A_2 \qquad\qquad \neg A_2 \rightarrow A_1,$$

that express the fact that we can compute the final value for A_2 (resp., A_1) by taking into account the prediction already made for A_1 (resp., A_2). The same intuition applies when we are given a requirement with n literals. Indeed, given (1), we can arbitrarily pick one of its literals, e.g., l_n, and consider the corresponding rule r:

$$\neg l_1 \wedge \neg l_2 \wedge \cdots \wedge \neg l_{n-1} \rightarrow l_n. \tag{2}$$

Expressions of the form (2) are *rules*, each representing the case in which we can infer the prediction of the *head* of the rule $head(r) = l_n$ to be true given that the literals in the *body* of the rule, $body(r) = \{\neg l_1, \neg l_2, \dots, \neg l_{n-1}\}$, have been predicted to be true. We anticipate that how the input set of requirements written in propositional logic is mapped into a corresponding set of rules can have an impact on a model's performance, as shown in the following.

Before presenting the general case, we first consider a basic case that allows us to highlight the main ideas of the proposed framework.

3.1. Basic case

We first outline our solution for a simple problem $(\mathcal{P}, \Pi)$ in which $\mathcal{A} = \{A_1, A_2\}$ and $\Pi = \{\neg A_1 \vee A_2\}$. Given $(\mathcal{P}, \Pi)$, we can either compute A_1 on the grounds of A_2, or compute A_2 on the grounds of A_1. As we will see in the rest of this section, this choice can have an impact on the performance, and we will explore both possibilities one by one.

Suppose that we want to compute A_2 on the grounds of A_1. Then, we can rewrite our unique requirement $\neg A_1 \vee A_2$ as the rule $A_1 \rightarrow A_2$. Following [13,5], we know that given a neural network h and a data point x, we can build ReqL by setting:

$$\begin{aligned} \text{ReqL}_{A_1} &= h_{A_1}, \\ \text{ReqL}_{A_2} &= \max(h_{A_2}, h_{A_1}). \end{aligned} \tag{3}$$

This guarantees that the requirement is always satisfied, however (again, as shown in [13,5]), if the layer is used together with the standard binary cross-entropy loss, this might cause wrong supervisions. We thus need a novel loss function, which we define as:

$$\begin{aligned} \text{ReqLoss}_{A_1} &= -y_{A_1} \ln(\text{ReqL}_{A_1}) - (1 - y_{A_1}) \ln(1 - \text{ReqL}_{A_1}), \\ \text{ReqLoss}_{A_2} &= -y_{A_2} \ln(\max(h_{A_2}, h_{A_1} y_{A_1})) - (1 - y_{A_2}) \ln(1 - \text{ReqL}_{A_2})). \end{aligned} \tag{4}$$

Notice that if $h_{A_1} y_{A_1} > h_{A_2} y_{A_2}$, the gradients of the ReqLoss_{A_2} will back-propagate through the A_1 output, and A_2 will be predicted correctly (assuming A_1 prediction is correct) given that the rule $(A_1 \rightarrow A_2)$ will be embedded in the constraint layer. Similarly to [13,5], ReqLoss and ReqL have been designed to have such behaviour on purpose, as this allows the network to learn how to delegate the prediction for one label (in this case A_2) to another (in this case A_1).

Suppose now that we want to compute A_1 on the grounds of A_2. Then, we can rewrite our unique requirement $\neg A_1 \vee A_2$ as the rule $\neg A_2 \rightarrow \neg A_1$, having as head $\neg A_1$ and as body $\{\neg A_2\}$. In this case, the models proposed in both [13] and [5] do not encompass this type of requirement. Indeed, the model proposed in [13] works only with hierarchical constraints, while the one proposed in [5] makes the assumption that the head of the rule must always be positive. Here, since we want to be able to capture any requirement expressed in propositional logic, we allow for the negation both in the head and in the body of each rule. Then, to map the requirements in our layer and loss, we need to define a negation function $f_\neg(\cdot)$ allowing us to associate a value $f_\neg(v)$ with $\neg A$ on the basis of the value v associated with the label A. The standard requirements for strict negation in fuzzy logics [17] are:

- $f_\neg(0) = 1$ and $f_\neg(1) = 0$,
- $f_\neg(\theta) = \theta$, and
- $f_\neg$ is strictly decreasing and continuous.

Indeed, these three requirements entail that for any value $v \in [0, 1]$, if $v < \theta$, then $f_\neg(v) > \theta$ and vice versa. Thus, for any model m and label A, if we extend m by defining $m(\neg A) = f_\neg(m(A))$, then exactly one between $m(A)$ and $m(\neg A)$ has a value higher than θ, matching our intuition that m predicts exactly one between A and $\neg A$. This result is a direct consequence of our assumption that a model m is not allowed to output the value θ as the prediction of a label A: without such assumption, we could have $m(A) = m(\neg A) = \theta$, contradicting the previously stated intuition. Given this, for any threshold θ, there are infinitely many functions $f_\neg$ satisfying the above three requirements. For simplicity, from here on, we consider the standard negation $f_\neg(v) = 1 - v$ for each $v \in [0, 1]$, which entails $\theta = 0.5$. All the definitions and results generalize to the case in which we have an arbitrary function respecting the three conditions for strict negation. Given the above, the requirements layer can be built in the following way:

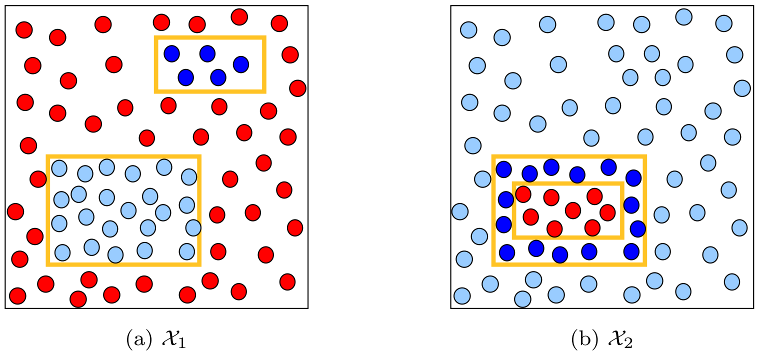


Fig. 2. Visualization of the two datasets $\mathcal{X}_1$ and $\mathcal{X}_2$. In both representations, the blue data points are associated with both classes A_1 and A_2, the light blue data points are associated only with the class A_2, while the red data points are associated with no classes. For visualization purposes, we do not plot all the data points, as the datasets consist of 5000 (50/50 train/test split) data points sampled from a uniform distribution over $[0,1]^2$. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

$$\begin{aligned} \mathrm{ReqL}_{A_1} &= 1-\max(1-h_{A_1}, 1-h_{A_2}) = \min(h_{A_1}, h_{A_2}) \\ \mathrm{ReqL}_{A_2} &= h_{A_2}. \end{aligned} \tag{5}$$

The loss is then updated accordingly:

$$\begin{aligned} \mathrm{ReqLoss}_{A_1} &= -y_{A_1}\ln(\mathrm{ReqL}_{A_1}) - (1-y_{A_1})\ln(1-\min(h_{A_1}, h_{A_2}(1-y_{A_2})) \\ \mathrm{ReqLoss}_{A_2} &= -y_{A_2}\ln(\mathrm{ReqL}_{A_2}) - (1-y_{A_2})\ln(1-\mathrm{ReqL}_{A_2}). \end{aligned} \tag{6}$$

Intuitively,

1. considering $A_1 \to A_2$ allows us to infer positive predictions for A_2 on the grounds of the positive predictions for A_1, and
2. considering $\neg A_2 \to \neg A_1$ allows us to infer negative predictions for A_1 on the grounds of the negative predictions for A_2,

and indeed we cannot consider both rules together in our framework in which either A_2 has to be computed based on A_1 or vice versa. To better understand the differences in considering either $A_1 \to A_2$ or $\neg A_2 \to \neg A_1$, we make use of two different datasets $\mathcal{X}_1$ and $\mathcal{X}_2$, which are visualized in Fig. 2, corresponding to the task of labeling points in a plane (thus, $D = 2$) knowing that it is always the case that $(\neg A_1 \vee A_2)$ holds.

For each dataset, we consider three different models:

1. a standard feedforward neural network f that does not incorporate any background knowledge about the problem,
2. the version of CCN$^+$ that computes A_2 on the grounds of A_1, which we call p–CCN$^+$, as it considers the requirement as a rule with positive head, and
3. the version of CCN$^+$ that computes $\neg A_1$ on the grounds of $\neg A_2$, which we call n–CCN$^+$, as it considers the requirement as a rule with negative head.

We implemented the above models in the following way. Regarding f, we implemented it as a feedforward neural network with one hidden layer with four neurons and *tanh* nonlinearity. We used the *sigmoid* non-linearity for the output layer. We then trained it using the binary cross-entropy loss with Adam optimization [18] for 20k epochs with learning rate 10^{-2} ($\beta_1 = 0.9$, $\beta_2 = 0.999$). We built p–CCN$^+$ (resp., n–CCN$^+$) by taking a neural network with exactly the same topology of f and adding the requirements layer defined in Equation (3) (resp., (5)). We then trained p–CCN$^+$ (resp., n–CCN$^+$) using the loss defined in Equation (4) (resp., (6)) again with Adam optimization for 20k epochs with learning rate 10^{-2} ($\beta_1 = 0.9$, $\beta_2 = 0.999$).

Consider first the dataset $\mathcal{X}_1$, corresponding to the left subfigure in Fig. 2a. We plot the decision boundaries obtained by each model in Fig. 3. Firstly, we notice that incorporating the background knowledge expressed by the requirement always leads to more fitting decision boundaries, no matter which rule corresponding to the requirement we consider: both p–CCN$^+$ and n–CCN$^+$ have a visibly better performance than the standard model f. If we focus on the differences between p–CCN$^+$ and n–CCN$^+$, we can see that the decision boundaries for A_2 created by p–CCN$^+$ are slightly better than the ones created by n–CCN$^+$. To understand the reasons behind this behaviour, we plotted the decision boundaries for each model *before* the requirements layer in Fig. 4. As we can see from the figure, p–CCN$^+$ is able to delegate the prediction for the label A_2 to the label A_1 for all those data points that fall in the smaller yellow rectangle. Such a breakdown of the more complex prediction into two simpler ones was not possible for n–CCN$^+$ given the structure of its requirements layer. However, n–CCN$^+$ was still able to use the requirement to learn a much simpler function for A_1 and delegate the prediction of $\neg A_1$ to $\neg A_2$ for all the data points surrounding the smaller rectangle. This, in turn, allowed the neural network to learn better decision boundaries for A_2. Thus, even in this case, using our requirements layer and loss led to more efficient utilization of the weights/network resources. The above considerations about the decision boundaries are reflected in the

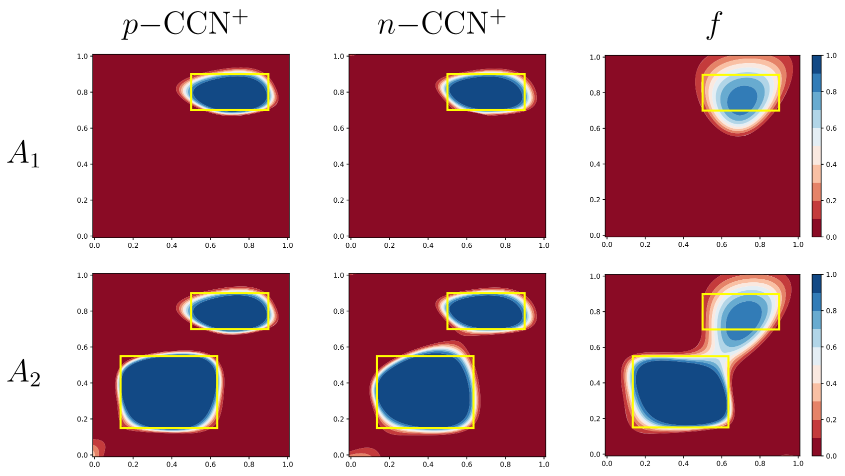


Fig. 3. Decision boundaries of the models $p-\mathrm{CCN}^+$, $n-\mathrm{CCN}^+$, and f for the dataset $\mathcal{X}_1$. In each figure, the darker the blue (resp., red), the more confident a model is that the data point belongs (does not belong) to the class (see scale at the end of each row).

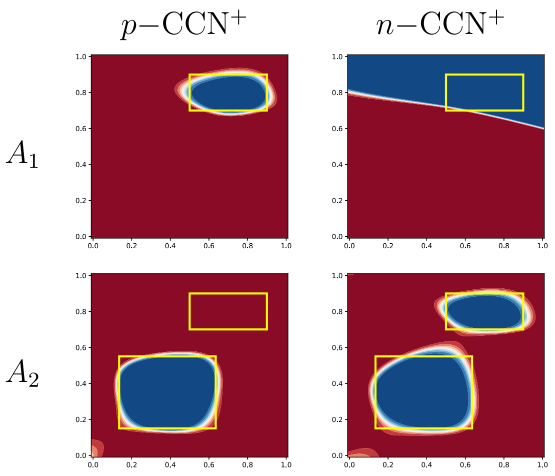


Fig. 4. Decision boundaries of the models $p-\mathrm{CCN}^+$ and $n-\mathrm{CCN}^+$ before the requirements layer for the dataset $\mathcal{X}_1$.

performance of each model measured in terms of accuracy. Indeed, we run each model with five different seeds, and we get the following results in terms of accuracy: (i) $p-\mathrm{CCN}^+$ gets 0.984 ± 0.006, (ii) $n-\mathrm{CCN}^+$ gets 0.963 ± 0.015, and (iii) f gets 0.941 ± 0.004.

Consider now the second dataset $\mathcal{X}_2$ corresponding to the right subfigure in Fig. 2b. We plot the decision boundaries obtained by each model in Fig. 5. In this case, we have that the decision boundaries produced by $n-\mathrm{CCN}^+$ are clearly much more fitting than those produced by $p-\mathrm{CCN}^+$ and f. This is due to its requirements layer, which allowed it to predict a simple rectangle for both A_1 and A_2, as shown in Fig. 6, and then compose the information to obtain the final prediction for A_2. If we focus on the differences between the boundaries produced by $p-\mathrm{CCN}^+$ and the ones produced by f, we can see that f violates the requirements for most of the data points in the small rectangle, as it predicts A_1 but not A_2, and it does not manage to capture the rectangular shapes, which are in turn approximated with a more triangular one. Thus, in this case also, we can see the benefits produced by the requirements layer. The above considerations done for the decision boundaries are again reflected in the performance of each model measured in terms of accuracy. Indeed, we run each model with five different seeds, and we get the following results for accuracy: (i) $p-\mathrm{CCN}^+$ gets 0.924 ± 0.012, (ii) $n-\mathrm{CCN}^+$ gets 0.966 ± 0.004, and (iii) f gets 0.915 ± 0.010.

As a final consideration for this basic case, please note that while in the datasets considered above it was always convenient for at least one of our models to learn to delegate some predictions, this might not always be the case. In such cases, our layer does not force in any way the delegation, and, for each datapoint, the model can learn whether to delegate or not the prediction for the head to the body. For a more detailed discussion on this point, please see [13], where we show that given a constraint $A_1 \to A_2$, the model can learn whether to delegate the prediction for A_2 to A_1 for: (i) all the points that belong to A_1, (ii) none of the points that belong to A_1 or (iii) a subset of the datapoints that belong to A_1.

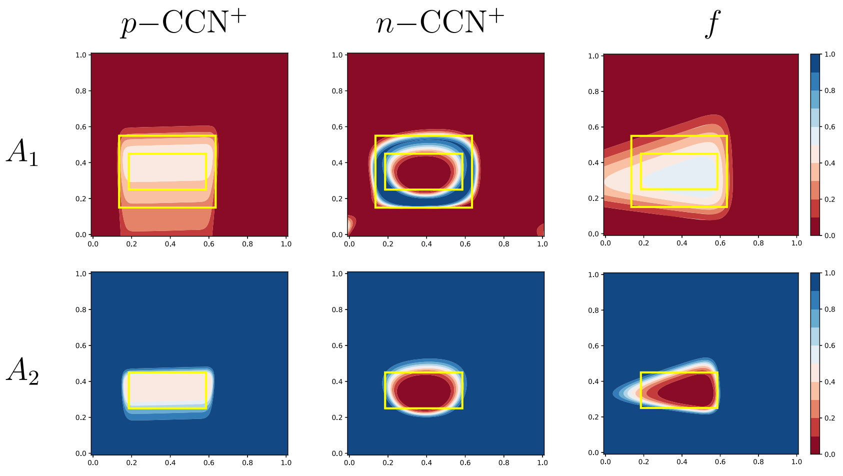


Fig. 5. Decision boundaries of models p−CCN$^+$, n−CCN$^+$ and f for dataset $\mathcal{X}_2$.

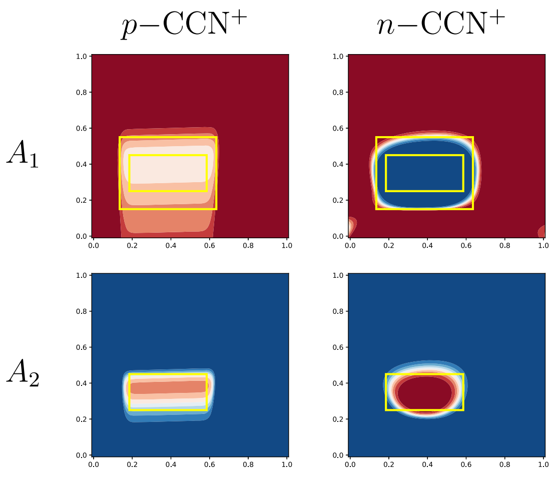


Fig. 6. Decision boundaries of the models p−CCN$^+$ and n−CCN$^+$ before the requirements layer for the dataset $\mathcal{X}_2$.

3.2. *General case*

In the previous section, we have seen that given a problem $\mathcal{P}$ with a single requirement with two literals, we can write the requirement in different ways, and that this choice will have an impact on the performance. In this section, we study what happens when $\mathcal{P}$ is equipped with multiple requirements, possibly with more than two literals. We first show how to build ReqL, and then we show how to define ReqLoss.

3.2.1. *Requirements layer*

We start with a problem $\mathcal{P}$ with a single requirement of the form:

$$l_1 \vee l_2 \vee \cdots \vee l_n \vee l. \tag{7}$$

Generalizing what we have done in the basic case, we can pick one of its literals, e.g., l, and consider the corresponding rule r:

$$\neg l_1 \wedge \neg l_2 \wedge \cdots \wedge \neg l_n \rightarrow l. \tag{8}$$

Given a neural network h for $\mathcal{P}$, and supposing that l is the label A, we can build ReqL by setting $\text{ReqL}_{A_1} = h_{A_1}, \ldots, \text{ReqL}_{A_n} = h_{A_n}$, and

$$\text{ReqL}_A = \max(h_A, \min(1 - h_{l_1}, 1 - h_{l_2}, \ldots, 1 - h_{l_n})), \tag{9}$$

where, if $l_i = \neg A_i$, $h_{\neg A_i}$ stands for $1 - h_{A_i}$ $(1 \le i \le n)$.[2] Equivalently, if $l = \neg A$, then we can build ReqL by setting $\text{ReqL}_{A_1} = h_{A_1}, \dots, \text{ReqL}_{A_n} = h_{A_n}$, and

$$\begin{aligned}\text{ReqL}_A &= 1 - \max(1 - h_A, \min(1 - h_{l_1}, 1 - h_{l_2}, \dots, 1 - h_{l_n})),\\ &= \min(h_A, \min(h_{l_1}, h_{l_2}, \dots, h_{l_n})).\end{aligned} \tag{10}$$

It can easily be seen that in both cases, ReqL_A ensures that the requirement is always satisfied.

Consider now the case in which we have multiple requirements involving the label A, and that A is also the label that we want to predict possibly on the basis of the other predictions, given the requirements. Assume, for simplicity, that every requirement either contains A or $\neg A$. In this simple scenario, $\mathcal{R}_A^+$ (resp., $\mathcal{R}_A^-$) represents the set of rules having the form (8) obtained by converting each clause in Π where A occurs positively (resp., negatively) into rules with head A (resp., $\neg A$). For the time being, we also suppose that the bodies of the rules in $\mathcal{R}_A^+$ and $\mathcal{R}_A^-$ are also mutually exclusive, i.e., that for any two rules $r^+ \in \mathcal{R}_A^+$ and $r^- \in \mathcal{R}_A^-$, there is a label A_1 such that $\{A_1, \neg A_1\} \subseteq body(r^+) \cup body(r^-)$. The rules in $\mathcal{R}_A^+$ and $\mathcal{R}_A^-$ define a *coherence interval* $[b_A^+, b_A^-]$ for A representing the set of admissible values for A, i.e., if A gets a value in $[b_A^+, b_A^-]$, then we are guaranteed the coherence with Π. The boundaries of this interval are:

$$b_A^+ = \max_{r \in \mathcal{R}_A^+}(\min_{l \in body(r)}(h_l)), \qquad b_A^- = 1 - \max_{r \in \mathcal{R}_A^-}(\min_{l \in body(r)}(h_l)).$$

Example 1. Let $\Pi = \{\neg A_1 \vee \neg A_2 \vee A, \neg A_3 \vee A, A_2 \vee A_3 \vee \neg A\}$. Then, $\mathcal{R}_A^+ = \{A_1 \wedge A_2 \to A, A_3 \to A\}$ and $\mathcal{R}_A^- = \{\neg A_2 \wedge \neg A_3 \to \neg A\}$. This entails $b_A^+ = \max(\min(h_{A_1}, h_{A_2}), h_{A_3})$ and $b_A^- = 1 - \min(1 - h_{A_2}, 1 - h_{A_3})$. Suppose then that the neural network returns $h_{A_1} = 0.6$, $h_{A_2} = 0.7$, and $h_{A_3} = 0.4$. Then, $b_A^+ = 0.6$ and $b_A^- = 0.7$. As we can see, if A is assigned any value between 0.6 and 0.7, then the prediction is guaranteed to satisfy the requirements.

Suppose now that we drop the assumption that the body of the rules in $\mathcal{R}_A^+$ and $\mathcal{R}_A^-$ are mutually exclusive. Then, the current definition of the bounds does not guarantee that the prediction for A satisfies the requirements.

Example 2. Let $\Pi = \{\neg A_1 \vee A, \neg A_2 \vee \neg A\}$. Then, $\mathcal{R}_A^+ = \{A_1 \to A\}$ and $\mathcal{R}_A^- = \{A_2 \to \neg A\}$. Further, suppose that the neural network outputs $h_{A_1} = h_{A_2} = 0.9$, and $h_A = 0.4$. According to our definition above, then we would have $b_A^+ = 0.9$, and $b_A^- = 0.1$, meaning that it is not possible to satisfy the requirements. Indeed, the problem is in the initial predictions for A_1 and A_2, which, independently from the prediction of A, lead to a violation of the requirements.

To avoid the above problem, we need to ensure that if the prediction satisfies the body of a rule in $\mathcal{R}_A^+$ (resp., $\mathcal{R}_A^-$), then there exists no rule in $\mathcal{R}_A^-$ (resp., $\mathcal{R}_A^+$) whose body is satisfied. To get an intuition on how to make this happen, we consider again the example above.

Example 3 *(Example 2, cont'd).* Given Π, we can apply the resolution rule and get the additional requirement $\neg A_1 \vee \neg A_2$, which in turn can be rewritten either as $A_1 \to \neg A_2$ or as $A_2 \to \neg A_1$. Considering the first rewriting, we can follow what we have done in the basic case, and we can define ReqL for A_1 and A_2 as:

$$\text{ReqL}_{A_1} = h_{A_1} = 0.9$$

$$\text{ReqL}_{A_2} = 1 - \max(1 - h_{A_2}, h_{A_1}) = 0.1.$$

Then, if we compute the bounds for A on the grounds of ReqL_{A_1} and ReqL_{A_2} instead of h_{A_1} and h_{A_2}, we obtain: $b_A^+ = \text{ReqL}_{A_1} = 0.9$ and $b_A^- = 1 - \text{ReqL}_{A_2} = 0.9$. If, on the other hand, we pick the second rewriting, we have:

$$\text{ReqL}_{A_1} = 1 - \max(1 - h_{A_1}, h_{A_2}) = 0.1$$

$$\text{ReqL}_{A_2} = h_{A_2} = 0.9,$$

and $b_A^+ = \text{ReqL}_{A_1} = 0.1$, $b_A^- = 1 - \text{ReqL}_{A_2} = 0.1$. Thus, no matter how $\neg A_1 \vee \neg A_2$ is converted into a rule, the requirements are now satisfied. Notice that starting from the original requirements $\Pi = \{\neg A_1 \vee A, \neg A_2 \vee \neg A\}$, we could have considered the rules $\{\neg A \to \neg A_1, A \to \neg A_2\}$ for which no problem would have arisen, and no resolution would have been needed.

As the example makes clear, in the process of updating the values of the labels in order to be coherent with the requirements, we may have also to consider the implicit requirements that can be derived from the input ones by resolution. Formally, given two clauses c_1 and c_2 such that there exists only one label, e.g., A, such that A in c_1 and $\neg A$ in c_2, the *resolution* $res(c_1, c_2)$ *of* c_1 *and* c_2 is

[2] Since, in this work, max and min take in input only values in $[0,1]$, from now on we define them as functions taking a set of values in $[0,1]$, and such that: $\max(\emptyset) = 0$, $\min(\emptyset) = 1$, and

$$\max(\{x\} \cup S) = \begin{cases} x, & \text{if } x \ge \max(S), \text{ and} \\ \max(S), & \text{otherwise,} \end{cases} \qquad \min(\{x\} \cup S) = \begin{cases} x, & \text{if } x \le \min(S), \text{ and} \\ \min(S), & \text{otherwise.} \end{cases}$$

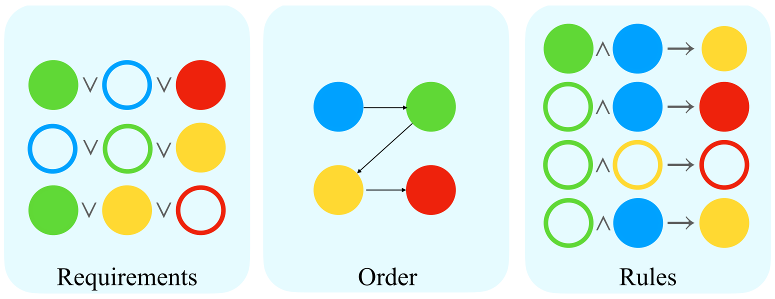


Fig. 7. Visualization of an example of how to obtain the rules from Π. In the visualization, each color corresponds to a label (thus, we have the labels "Green", "Blue", "Red", and "Yellow"), and if a label occurs negatively, then its circle appears empty.

the clause obtained by removing the disjuncts A and $\neg A$ from $(c_1 \vee c_2)$. Then, starting from some initial prediction possibly violating the requirements (in the example, $h_{A_1} = h_{A_2} = 0.9$, $h_A = 0.4$), (i) some of the initial values (in the example, $\text{ReqL}_{A_1} = 0.9$) are used to update the value of other labels because of the requirements (in the example, ReqL_{A_2}), and (ii) the new values are taken into account in order to fix the values for the other labels (in the example, ReqL_A).

In general, the output for each label is computed sequentially, starting from the label at level 0 (whose value corresponds to the one computed by the neural network) up to the last label at level $|\mathcal{A}| - 1$. Formally, we associate to each label A an integer in $[0, |\mathcal{A}| - 1]$. This integer represents the *level* $\lambda(A)$ of the label $A \in \mathcal{A}$, where $\lambda : \mathcal{A} \mapsto [0, |\mathcal{A}| - 1]$ is a *label ordering* function, assumed, for the time being, to be surjective. In Section 4, we will discuss how to drop the surjectivity assumption in order to optimize the computation on GPUs, while in Section 5, we will show how to choose label orderings that lead to better performance.

Given a label ordering, we associate with each label A a set Π_A of requirements inductively defined as follows:

1. if A is the label at the highest level $|\mathcal{A}| - 1$, $\Pi_A = \Pi$, and
2. if A is a label at level $k < |\mathcal{A}| - 1$, and A_1 is the label at level $k + 1$, $\Pi_A = \Pi_{A_1} \setminus (\Pi^+_{A_1} \cup \Pi^-_{A_1}) \cup \{res(c_1, c_2) : c_1 \in \Pi^+_{A_1}, c_2 \in \Pi^-_{A_1}\}$.

In practice, starting from Π associated with the label at level $|\mathcal{A}| - 1$, the set of requirements associated with the label at each level (lower than $|\mathcal{A}| - 1$) is obtained from the upper one by

1. removing the requirements with either the positive or negative label associated with the level, and
2. adding the requirements that can be derived by resolution over the label corresponding to the level.

In practice, we are bucket partitioning the input set of clauses as described in [19], which also proves that the construction is exponentially bounded (in time and space) in the induced width of the theory's interaction graph in which a node is associated with a proposition and an arc connects any two nodes appearing in the same clause (for more details, see [19]). Clearly, the set of requirements associated with the label at level 0, e.g., A, is either empty or contains a single requirement that can be A or $\neg A$. Notice also that we do not necessarily consider all the possible resolutions between the requirements in Π. For instance, if $\Pi = \{A_1 \vee \neg A, A \vee A_2\}$, the clause $A_1 \vee A_2$ will be computed only if the label associated with level 2 is A. For each label A, we can then extract from Π_A (i) the set Π^+_A of clauses in Π_A where A occurs positively, and (ii) the set Π^-_A of clauses in Π_A where A occurs negatively. Then, given the set of requirements Π^+_A (resp., Π^-_A), we can define the corresponding set of rules $\mathcal{R}^+_A$ (resp., $\mathcal{R}^-_A$) obtained by converting each clause in Π^+_A (resp., Π^-_A) into rules with head A (resp., $\neg A$). A visualization of an example of how we obtain the rules from the initial set of requirements is given in Fig. 7, in which the set of requirements

$$(\text{Green} \vee \neg\text{Blue} \vee \text{Red}), \quad (\neg\text{Blue} \vee \neg\text{Green} \vee \text{Yellow}), \quad (\text{Green} \vee \text{Yellow} \vee \neg\text{Red})$$

given the variable ordering

$$\text{Blue}, \text{Green}, \text{Yellow}, \text{Red}$$

is converted into the set of rules

$$(\neg\text{Green} \wedge \text{Blue} \rightarrow \text{Red}), \quad (\neg\text{Green} \wedge \neg\text{Yellow} \rightarrow \neg\text{Red}), \quad (\text{Green} \wedge \text{Blue} \rightarrow \text{Yellow}), \quad (\neg\text{Green} \wedge \text{Blue} \rightarrow \text{Yellow}).$$

For each label A, the value ReqL_A is then defined by

$$\text{ReqL}_A = \min(\max(h_A, b^+_A), b^-_A), \tag{11}$$

where

$$b^+_A = \max_{r \in \mathcal{R}^+_A}(\min_{l \in body(r)}(\text{ReqL}_l)), \qquad b^-_A = 1 - \max_{r \in \mathcal{R}^-_A}(\min_{l \in body(r)}(\text{ReqL}_l)), \tag{12}$$

and for any label A, $\mathrm{ReqL}_{\neg A} = 1 - \mathrm{ReqL}_A$.

The definition of ReqL_A is well-founded given that

1. if A is at level $i \in (0, |\mathcal{A}| - 1]$, Π_A contains only labels at levels $\leq i$, and thus the bodies of the rules in $\mathcal{R}^+_A$ and $\mathcal{R}^-_A$ contain only literals whose label has level $< i$, and
2. if A is at level 0, Π_A is either empty, and in this case $\mathrm{ReqL}_A = h_A$ (and $\mathrm{ReqL}_{\neg A} = 1 - h_A$) or $\Pi_A = \{A\}$ (resp., $\Pi_A = \{\neg A\}$) in which case $\mathrm{ReqL}_A = 1$ (resp., $\mathrm{ReqL}_A = 0$).

Given the above definition, we can prove that for any MC problem with requirements $(\mathcal{P}, \Pi)$, any neural network h for $\mathcal{P}$ and any label ordering, all predictions made by CCN^+ are guaranteed to be compliant with Π.

Theorem 1. *Let $(\mathcal{P}, \Pi)$ be an MC problem with propositional requirements. Let h be an MC model for $\mathcal{P}$. Let λ be a label ordering. Then, the model* CCN^+ *built on top of h on the basis of λ is coherent with* Π.

Proof. The proof is by induction on the number n of labels in Π.

For the base case ($n = 0$), $\Pi = \emptyset$, and for any label A, $\mathrm{ReqL}_A = h_A$, and the statement trivially holds.

Assume that there are $n = k + 1$ labels. Assume that A is the label in Π associated with the highest level and that A_1 is the label associated with the preceding level. Then, $\Pi_A = \Pi$ and $\Pi_{A_1} = \Pi_A \setminus (\Pi^+_A \cup \Pi^-_A) \cup \{res(c_1, c_2) : c_1 \in \Pi^+_A, c_2 \in \Pi^-_A\}$. Π_{A_1} has less than n labels and thus, by the induction hypothesis, CCN^+ is coherent by construction with Π_{A_1}. To prove the statement, it is thus sufficient to show that ReqL satisfies all the clauses in Π^+_A and Π^-_A. From the inductive hypothesis, we know that ReqL satisfies Π_{A_1}, and thus it also satisfies $\{res(c_1, c_2) : c_1 \in \Pi^+_A, c_2 \in \Pi^-_A\}$.

Assume that there is a rule $r \in \mathcal{R}^-_A$ that is violated. Then, it holds that $\min_{l \in body(r)}(\mathrm{ReqL}_l) > \theta$, entailing $b^-_A < \theta$, and $\mathrm{ReqL}_{\neg A} < \theta$, and thus $\mathrm{ReqL}_A > \theta$. However, it is not possible to have $b^-_A < \theta$ and $\mathrm{ReqL}_A > \theta$ given the definition of ReqL_A.

Assume that there is a rule $r_1 \in \mathcal{R}^+_A$ that is violated. Then, it holds that $\min_{l \in body(r_1)}(\mathrm{ReqL}_l) > \theta$ and $\mathrm{ReqL}_A < \theta$, and thus $b^+_A > \theta$ and $b^-_A < \theta$. From $b^-_A < \theta$, it follows that there exists a rule $r_2 \in \mathcal{R}^-_A$ such that $\min_{l \in body(r_2)}(\mathrm{ReqL}_l) > \theta$. Consider the clauses $c_1 \in \Pi^+_A$ and $c_2 \in \Pi^-_A$ corresponding to r_1 and r_2. There are two cases, both contradicting $\min_{l \in body(r_1)}(\mathrm{ReqL}_l) > \theta$ and $\min_{l \in body(r_2)}(\mathrm{ReqL}_l) > \theta$:

1. either there exists a label $A_2 \neq A$ such that $\{A_2, \neg A_2\} \subseteq body(r_1) \cup body(r_2)$, and we know that $\mathrm{ReqL}_{A_2} > \theta$ if and only if $\mathrm{ReqL}_{\neg A_2} < \theta$,
2. or there does not exist such label, which implies that $res(c_1, c_2) \in \Pi_{A_1}$. However, this is not possible, since by the induction hypothesis, ReqL satisfies Π_{A_1} and thus also $res(c_1, c_2)$. □

While the above formulation guarantees that all predictions are coherent with Π, it might be the case that for some label A, we have that $b^+_A > b^-_A$, which in turn implies that:

$$\min(\max(h_A, b^+_A), b^-_A) \neq \max(\min(h_A, b^-_A), b^+_A).$$

This might happen for the labels in the head of rules whose body are independently computed one from the others.

Example 4. Let $\Pi = \{A_1 \vee \neg A_2 \vee \neg A, \neg A_3 \vee A\}$, and assume that A has the level 3, A_1 has the level 2, A_2 has the level 1, and A_3 has the level 0. The set of requirements associated with each level are:

$$\Pi_A = \Pi, \qquad \Pi_{A_1} = \{A_1 \vee \neg A_2 \vee \neg A_3\}, \qquad \Pi_{A_2} = \emptyset, \qquad \Pi_{A_3} = \emptyset.$$

Further, let $h_A = 0.7$, $h_{A_1} = 0.4$, $h_{A_2} = 0.8$, and $h_{A_3} = 0.9$. Then, we get $\mathrm{ReqL}_{A_3} = h_{A_3} = 0.9$, $\mathrm{ReqL}_{A_2} = h_{A_2} = 0.8$,

$$\mathrm{ReqL}_{A_1} = \min(\max(h_{A_1}, b^+_{A_1}), b^-_{A_1}) = \max(\min(h_{A_1}, b^-_{A_1}), b^+_{A_1}) = 0.8.$$

However, for the label A, we obtain:

$$b^+_A = \mathrm{ReqL}_{A_3} = 0.9 \qquad\qquad b^-_A = 1 - \min(\mathrm{ReqL}_{A_2}, 1 - \mathrm{ReqL}_{A_1}) = 0.8,$$

which shows that in this case, $\min(\max(h_A, b^+_A), b^-_A) \neq \max(\min(h_A, b^-_A), b^+_A)$.

Notice that even in the case $b^+_A > b^-_A$, Theorem 1 holds despite the fact that the coherence interval $[b^+_A, b^-_A]$ in this case is empty. The theorem indeed guarantees that for each rule r, if $\min_{l \in body(r)} \mathrm{ReqL}_l > \theta$, then $\mathrm{ReqL}_{head(r)} > \theta$, but does not guarantee the stronger condition that

$$\min_{l \in body(r)} \mathrm{ReqL}_l \leq \mathrm{ReqL}_{head(r)}.$$

When a model satisfies this stronger condition for each rule corresponding to a requirement in Π, then we say that the model is *strongly coherent* with Π. Strong coherence entails coherence, and is ensured if for every label A, both $\mathcal{R}^+_A$ and $\mathcal{R}^-_A$ are saturated,

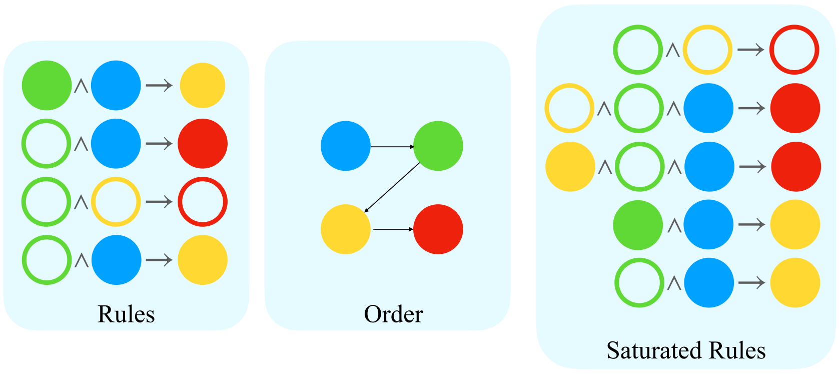


Fig. 8. Visualization of an example of how to obtain saturated rules.

i.e., if for each rule $r \in \mathcal{R}^+_A \cup \mathcal{R}^-_A$, the label at the highest level occurring in the body of a rule in $\mathcal{R}^+_A \cup \mathcal{R}^-_A$ occurs also in r. More formally, given a label ordering λ, a label B is the *saturating atom of label A (with respect to λ)* if there exists a rule $r \in \mathcal{R}^+_A \cup \mathcal{R}^-_A$ such that B occurs in the body of r (i.e., $\{B, \neg B\} \cap body(r) \neq \emptyset$), and for each rule $r \in \mathcal{R}^+_A \cup \mathcal{R}^-_A$, there does not exist a label C occurring in the body of r with $\lambda(C) > \lambda(B)$. The set Π is *saturated (with respect to λ)* if for each label $A \in \mathcal{A}$ and for each rule $r \in \mathcal{R}^+_A \cup \mathcal{R}^-_A$, the saturating label of A occurs in the body of r. Notice that given a non-saturated set of rules $\mathcal{R}^+_A$ (resp., $\mathcal{R}^-_A$), it is always possible to build the equivalent saturated set of rules by taking the saturating atom B for A and replacing every rule $r \in \mathcal{R}^+_A$ (resp., $r \in \mathcal{R}^-_A$) in which B does not occur with the two rules obtained from r by adding once B and once $\neg B$ to its body. A visualization of an example on how to obtain the saturated rules is given in Fig. 8. Analogously, if the set Π of requirements is not saturated, it is always possible to compute the equivalent saturated set of requirements simply by converting back to requirements the saturated set of rules obtained from Π using the described saturation procedures for rules.

Consider again Example 4. We show that if we consider the saturated set of rules corresponding to Π, we obtain

1. $b^+_A \leq b^-_A$,
2. $\min(\max(h_A, b^+_A), b^-_A) = \max(\min(h_A, b^-_A), b^+_A)$, and
3. strong coherence of the model.

Example 5 *(Example 4 cont'd).* Given $\Pi = \{A_1 \vee \neg A_2 \vee \neg A, \neg A_3 \vee A\}$ and the levels as defined in Example 4, if we compute the set of requirements associated with each label, we get as before:

$$\Pi_A = \Pi, \qquad \Pi_{A_1} = \{A_1 \vee \neg A_2 \vee \neg A_3\}, \qquad \Pi_{A_2} = \emptyset, \qquad \Pi_{A_3} = \emptyset.$$

However, since we want that each set of rules is saturated, for $\mathcal{R}^+_A$, we now get:

$$\mathcal{R}^+_A = \{A_1 \wedge A_3 \to A, \neg A_1 \wedge A_3 \to A\},$$

while $\mathcal{R}^-_A = \{\neg A_1 \wedge A_2 \to \neg A\}$, as before. With these rules, we get again, $\text{ReqL}_{A_3} = h_{A_3} = 0.9$, $\text{ReqL}_{A_2} = h_{A_2} = 0.8$, and $\text{ReqL}_{A_1} = 0.8$. Regarding the label A, the boundaries and final value are:

$$b^+_A = \max(\min(\text{ReqL}_{A_1}, \text{ReqL}_{A_3}), \min(1 - \text{ReqL}_{A_1}, \text{ReqL}_{A_3})) = 0.8,$$

$$b^-_A = 1 - \min(1 - \text{ReqL}_{A_1}, \text{ReqL}_{A_2}) = 0.8,$$

$$\text{ReqL}_A = \min(\max(h_A, b^+_A), b^-_A) = \max(\min(h_A, b^+_A), b^-_A) = 0.8.$$

This example shows why saturating $\mathcal{R}^+_A$ and $\mathcal{R}^-_A$ is needed to obtain strong coherence. Indeed, while in the Boolean domain the two formulas $A_3 \to A$ and $(A_1 \wedge A_3 \to A) \wedge (\neg A_1 \wedge A_3 \to A)$ are equivalent, it is not the case in the continuous case. Rewriting the requirement as described above allows us to express b^+_A not only as a function of ReqL_{A_3}, but also as a function of ReqL_{A_1}. Consequently, if $\text{ReqL}_{A_3} \geq \max(1 - \text{ReqL}_{A_2}, \text{ReqL}_{A_1})$, as in Example 4, we have that $\text{ReqL}_{A_3} \geq \text{ReqL}_{A_1}$ and $\text{ReqL}_{A_3} \geq 1 - \text{ReqL}_{A_2}$, entailing $b^+_A = \max(\text{ReqL}_{A_1}, 1 - \text{ReqL}_{A_1})$ and $b^-_A \geq \text{ReqL}_{A_1}$ and $b^-_A \geq 1 - \text{ReqL}_{A_2} \geq 1 - \text{ReqL}_{A_1}$.

We now prove that CCN$^+$ is strongly coherent with the saturated set of requirements equivalent to Π.

Theorem 2. *Let $(\mathcal{P}, \Pi)$ be an MC problem with propositional requirements. Let h be an MC model for $\mathcal{P}$. Let λ be a label ordering. Then, the model CCN$^+$ built on top of h on the basis of λ is strongly coherent with the saturated set of requirements equivalent to Π.*

Proof. For notational convenience, we assume that Π is saturated (otherwise we consider the saturated set Π' of requirement equivalent to Π and substitute Π with Π' in the rest of the proof). Proving the strong coherence of CCN$^+$ amounts to prove that for any rule r corresponding to a requirement in Π,

$$\mathrm{ReqL}_{head(r)} \geq \min_{l \in body(r)} (\mathrm{ReqL}_l).$$

The proof is by induction on the level i of the head l of the rule r.

For the base case $i = 0$, $\Pi = \emptyset$ or $\Pi = \{l\}$, and the statement easily follows.

For the step case $i = j + 1$, if $l = \neg A$, then $r \in \mathcal{R}^-_A$, and the statement follows from the definition of ReqL at Equation (11). Indeed,

$$\begin{aligned}\mathrm{ReqL}_{\neg A} &= 1 - \mathrm{ReqL}_A \\ &= 1 - \min(\max(h_A, b^+_A), b^-_A) \\ &\geq 1 - b^-_A \\ &= \max\nolimits_{r \in \mathcal{R}^-_A}(\min\nolimits_{l \in body(r)}(\mathrm{ReqL}_l)) \\ &\geq \min\nolimits_{l \in body(r)}(\mathrm{ReqL}_l).\end{aligned}$$

Consider the case in which the rule r has a positive head A, i.e., $l = A$ and thus $r \in \mathcal{R}^+_A$.

Assume by contradiction that the thesis does not hold, i.e., that

$$\min_{l \in body(r)} (\mathrm{ReqL}_l) > \mathrm{ReqL}_A.$$

Then, this is possible only if $\mathrm{ReqL}_A = b^-_A$, i.e., there exists a rule $r^- \in \mathcal{R}^-_A$ such that:

$$\mathrm{ReqL}_A = 1 - \min_{l \in body(r^-)} (\mathrm{ReqL}_l)$$

and thus,

$$\min_{l \in body(r)} (\mathrm{ReqL}_l) > 1 - \min_{l \in body(r^-)} (\mathrm{ReqL}_l). \tag{13}$$

Assume that there exists a label B (which can be the saturating atom for A) such that $\{B, \neg B\} \subseteq body(r) \cup body(r^-)$. Then, considering the case in which $\neg B \in body(r)$ (the case $B \in body(r)$ is analogous)

$$\min_{l \in body(r)} (\mathrm{ReqL}_l) \leq 1 - \mathrm{ReqL}_B, \qquad \min_{l \in body(r^-)} (\mathrm{ReqL}_l) \leq \mathrm{ReqL}_B,$$

and from the two statements it follows

$$\min_{l \in body(r)} (\mathrm{ReqL}_l) \leq 1 - \min_{l \in body(r^-)} (\mathrm{ReqL}_l),$$

which contradicts (13).

We are now left with the case in which $body(r) \cup body(r^-)$ is consistent, and thus, if B is the saturating label for A either $B \in body(r) \cap body(r^-)$ or $\neg B \in body(r) \cap body(r^-)$. By construction, the level k of B is lower than i. For simplicity, we consider only the case $B \in body(r) \cap body(r^-)$ (the other case is analogous).

Condition (13) can be written as

$$\min(\mathrm{ReqL}_B, \min_{l \in body(r) \setminus \{B\}} (\mathrm{ReqL}_l)) > 1 - \min(\mathrm{ReqL}_B, \min_{l \in body(r^-) \setminus \{B\}} (\mathrm{ReqL}_l)).$$

With the abbreviation $\mathcal{L}^+ = body(r) \setminus \{B\}$ and $\mathcal{L}^- = body(r^-) \setminus \{B\}$, the above equation can be rewritten as:

$$\min(\mathrm{ReqL}_B, \min_{l \in \mathcal{L}^+} (\mathrm{ReqL}_l)) + \min(\mathrm{ReqL}_B, \min_{l \in \mathcal{L}^-} (\mathrm{ReqL}_l)) > 1. \tag{14}$$

Assume that $\min_{l \in \mathcal{L}^+ \cup \mathcal{L}^-} (\mathrm{ReqL}_l) = \min_{l \in \mathcal{L}^-} (\mathrm{ReqL}_l)$ (analogous considerations hold if $\min_{l \in \mathcal{L}^+ \cup \mathcal{L}^-} (\mathrm{ReqL}_l) = \min_{l \in \mathcal{L}^+} (\mathrm{ReqL}_l)$), which entails

$$\min_{l \in \mathcal{L}^-} (\mathrm{ReqL}_l) \leq \min_{l \in \mathcal{L}^+} (\mathrm{ReqL}_l). \tag{15}$$

Since B is the saturating atom of A, B is the label at the highest level occurring in $body(r) \cup body(r^-)$ and there must be a rule at level k with body $\mathcal{L}^+ \cup \mathcal{L}^-$ and head $\neg B$. Then, from the inductive hypothesis:

$$\mathrm{ReqL}_{\neg B} \geq \min_{l \in \mathcal{L}^+ \cup \mathcal{L}^-} (\mathrm{ReqL}_l),$$

entailing

$$1 - \mathrm{ReqL}_B \geq \min_{l \in \mathcal{L}^+ \cup \mathcal{L}^-} (\mathrm{ReqL}_l) = \min_{l \in \mathcal{L}^-} (\mathrm{ReqL}_l).$$

Consider the two cases:

1. $\min_{l\in\mathcal{L}^+}(\mathrm{ReqL}_l) \le \mathrm{ReqL}_B$. Then, (from (15)), $\min_{l\in\mathcal{L}^-}(\mathrm{ReqL}_l) \le \mathrm{ReqL}_B$, and thus from (14):
$$\min_{l\in\mathcal{L}^+}(\mathrm{ReqL}_l) + \min_{l\in\mathcal{L}^-}(\mathrm{ReqL}_l) > 1,$$
which, given $\min_{l\in\mathcal{L}^+}(\mathrm{ReqL}_l) \le \mathrm{ReqL}_B$, entails $1-\mathrm{ReqL}_B < \min_{l\in\mathcal{L}^-}(\mathrm{ReqL}_l)$, and this contradicts the inductive hypothesis.
2. $\min_{l\in\mathcal{L}^+}(\mathrm{ReqL}_l) > \mathrm{ReqL}_B$. Then, from (14):
$$\mathrm{ReqL}_B + \min_{l\in\mathcal{L}^-}(\mathrm{ReqL}_l) \ge \mathrm{ReqL}_B + \min(\mathrm{ReqL}_B, \min_{l\in\mathcal{L}^-}(\mathrm{ReqL}_l)) > 1,$$
entailing $1-\mathrm{ReqL}_B < \min_{l\in\mathcal{L}^-}(\mathrm{ReqL}_l)$ and thus contradicting the inductive hypothesis. □

Corollary 2.1. *In the hypotheses of Theorem 2, for each label A*

1. *the coherence interval* $[b_A^+, b_A^-]$ *for A is not empty, and*
2. $\min(\max(h_A, b_A^+), b_A^-) = \max(\min(h_A, b_A^-), b_A^+)$.

Proof. We consider each statement separately.

1. Assume that $b_A^+ > b_A^-$. Then, $\mathrm{ReqL}_A = b_A^-$, and since $b_A^- < 1$, let $r^- \in \mathcal{R}_A^-$ be the rule such that
$$b_A^- = 1 - \min_{l\in body(r^-)}(\mathrm{ReqL}_l),$$
and, since $b_A^+ > 0$, let $r^+ \in \mathcal{R}_A^+$ be the rule such that
$$b_A^+ = \min_{l\in body(r^+)}(\mathrm{ReqL}_l).$$
Then, given $\mathrm{ReqL}_A = b_A^-$, and the strong coherence of CCN$^+$,
$$\mathrm{ReqL}_A \ge \min_{l\in body(r^+)}(\mathrm{ReqL}_l) > 1 - \min_{l\in body(r^-)}(\mathrm{ReqL}_l) = \mathrm{ReqL}_A\,,$$
which is not possible.
2. The last statement is an easy consequence of the previous. □

3.2.2. Requirements loss

Once we have defined ReqL such that it guarantees that the requirements are always satisfied, we now modify the binary cross-entropy loss in order to ensure the supervisions on labels agree with their ground truth.

Consider an MC problem $\mathcal{P}$ and a neural network h for $\mathcal{P}$.

The binary cross-entropy loss $\mathcal{L}$ for a data point $(x, \mathcal{Y})$ is defined as:
$$\mathcal{L} = \sum_{A\in\mathcal{A}} \mathcal{L}_A,$$
$$\mathcal{L}_A = -y_A \log(h_A) - (1-y_A)\log(1-h_A),$$
where (i) $y_A = 1$ if $A \in \mathcal{Y}$, and $y_A = 0$ otherwise, and (ii) h_A represents the prediction made by the model h for the label A.

As shown in [13,5], if we use the layer ReqL with the binary cross-entropy loss, we might obtain supervisions disagreeing with their respective ground truth.

Example 6. Assume that $\Pi = \{\neg A_1 \vee A_2 \vee A\}$ and that
$$\mathrm{ReqL}_{A_1} = h_{A_1}, \qquad \mathrm{ReqL}_{A_2} = h_{A_2}, \qquad \mathrm{ReqL}_A = \max(h_A, \min(h_{A_1}, 1-h_{A_2})).$$
Then, considering the label A,
$$\mathcal{L}_A = -y_A \log(\mathrm{ReqL}_A) - (1-y_A)\log(1-\mathrm{ReqL}_A).$$
Suppose that $y_{A_1} = 0$, $y_{A_2} = 0$, $y_A = 1$, $h_{A_1} = 0.4$, $h_{A_2} = 0.3$, and $h_A = 0.2$. Then,
$$\mathcal{L}_A = -\log(\max(h_A, \min(h_{A_1}, 1-h_{A_2}))) = -\log(0.4)$$
$$\frac{\partial \mathcal{L}_A}{\partial h_{A_1}} = -\frac{1}{h_{A_1}} = -2.5\,.$$
Thus, teaching the neural network h to increase the output for the label A_1 even though $y_{A_1} = 0$.

As this example shows, the application of the standard cross-entropy loss may result in wrong supervisions given to some labels. As in [13,5], the problem is due to the fact that when selecting the label to which apply the supervision, its ground truth is not taken into account.

In general, given a data point $(x, \mathcal{Y})$ and a label A with ground truth $y_A = 1$, in determining the label to which we apply the supervisions, we have to take into account

1. The rules in $\mathcal{R}^+_A$ whose literals in the body are satisfied by the ground truth $\mathcal{Y}$: these rules are exploited to learn the left bound b^+_A for ReqLA, which will be determined by
$$\max_{r\in\mathcal{R}^+_A}(\min_{l\in body(r)}(y_l \mathrm{ReqL}_l)), \tag{16}$$
where y_l stands for $(1-y_B)$ when $l = \neg B$, $B \in \mathcal{A}$. Notice that if there are no rules whose body is satisfied by the ground truth, the above expression will correctly return 0 as lower bound.
2. The rules in $\mathcal{R}^-_A$ whose body surely contains a literal that is not satisfied by the ground truth $\mathcal{Y}$: these rules are exploited to learn the right bound b^-_A for ReqLA, which will be determined by
$$1-\max_{r\in\mathcal{R}^-_A}(\min_{l\in body(r)}(y_l + (1-y_l)\mathrm{ReqL}_l)). \tag{17}$$
Notice that if there are no rules in $\mathcal{R}^-_A$, the above expression will correctly return 1 as upper bound, while if $\mathcal{R}^-_A$ is not empty, it will return a value of the form $1-\mathrm{ReqL}_l$ in which l is a literal not satisfied by $\mathcal{Y}$.
3. The value of h_A computed by the neural networks for the label A.

Analogous considerations can be done for the case in which $y_A = 0$. Given that

$$\mathrm{ReqL}_A = \min(\max(h_A, b^+_A), b^-_A),$$

we define

$$\begin{array}{r}\mathrm{ReqLoss} = \sum_{A\in\mathcal{A}} \mathrm{ReqLoss}_A,\\ \mathrm{ReqLoss}_A = -y_A \log(\mathrm{ReqL}^p_A) - (1-y_A)\log(1-\mathrm{ReqL}^n_A),\end{array} \tag{18}$$

where

$$\begin{array}{l}\mathrm{ReqL}^p_A = \min(\ \max(h_A, \max_{r\in\mathcal{R}^+_A}(\min_{l\in body(r)}(y_l \mathrm{ReqL}_l))),\\ \qquad\qquad 1-\max_{r\in\mathcal{R}^-_A}(\min_{l\in body(r)}(y_l + (1-y_l)\mathrm{ReqL}_l)))\end{array}$$

is obtained by substituting (16) and (17) for b^+_A and b^-_A respectively, in the above definition of ReqL$_A$. Similarly, for the case in which $y_A = 0$,

$$\begin{array}{l}\mathrm{ReqL}^n_A = \min(\ \max(h_A, \max_{r\in\mathcal{R}^+_A}(\min_{l\in body(r)}(y_l + (1-y_l)\mathrm{ReqL}_l))),\\ \qquad\qquad 1-\max_{r\in\mathcal{R}^-_A}(\min_{l\in body(r)}(y_l \mathrm{ReqL}_l))).\end{array}$$

We now show that ReqLoss allows us to give the right supervisions even in the case illustrated in the example above.

Example 7 *(Example 6, cont'd).* Suppose that we have the same output and ground truth as in Example 6. Then, for A_1 and A_2, we have:

$$\mathrm{ReqLoss}_{A_1} = \mathcal{L}_{A_1} = -\log(0.6) \qquad \mathrm{ReqLoss}_{A_2} = \mathcal{L}_{A_2} = -\log(0.7).$$

Consider the loss for the label A. In this case, given the unique rule $A_1 \wedge \neg A_2 \to A$, we have $\min_{l\in body(r)} y_l = 0$ and $y_{head(r)} = 1$, and thus the loss:

$$\begin{aligned}\mathrm{ReqLoss}_A &= -\log(\mathrm{ReqL}^p_A)\\ &= -\log(\min(\max(h_A, \min(y_{A_1}h_{A_1}, (1-y_{A_2})(1-h_{A_2}))), 1))\\ &= -\log(\max(0.2, \min(0, 0.7)))\\ &= -\log(0.2).\end{aligned}$$

We obtain $\partial\mathrm{ReqLoss}_A/\partial h_A = -5.0$, which correctly teaches the model to increase the output for the label A.

Theorem 3. *Let $(\mathcal{P}, \Pi)$ be an MC problem with requirements. For any model h for $\mathcal{P}$ and class A, let $\frac{\partial \mathrm{ReqLoss}}{\partial h_A}$ be the partial derivative of* ReqLoss *with respect to h_A. For each data point, if $A \notin \mathcal{Y}$, then $\frac{\partial \mathrm{ReqLoss}}{\partial h_A} \geq 0$, and if $A \in \mathcal{Y}$, then $\frac{\partial \mathrm{ReqLoss}}{\partial h_A} \leq 0$.*

Proof. Let h be a model for $\mathcal{P}$. Consider a data point and a class A. The partial derivative of ReqLoss with respect to h_A can be computed as:

$$\frac{\partial \text{ReqLoss}}{\partial h_A} = \sum_{A \in \mathcal{A}} \frac{\partial \text{ReqLoss}_A}{\partial h_A}.$$

Let $y_A = 1$ (analogous considerations hold for $y_A = 0$).

Consider a class $B \in \mathcal{A}$.

1. Suppose $y_B = 1$. Then,

$$\text{ReqLoss}_B = -\log(\text{ReqLoss}^p_B) \quad \text{and} \quad \frac{\partial \text{ReqLoss}_B}{\partial h_A} \leq 0.$$

This is due to the fact that:

- either $\text{ReqLoss}_B = h_A$, which is possible only if $A = B$ or there exists a rule $r \in \mathcal{R}^+_B$ such that $A \in body(r)$ whose body is satisfied by the ground truth and:

$$\max_{r \in \mathcal{R}^+_B}(\min_{l \in body(r)}(y_l \text{ReqL}_l)) = h_A.$$

Then,

$$\frac{\partial \text{ReqLoss}_B}{\partial h_A} = -\frac{1}{h_A} \leq 0,$$

- or $\text{ReqLoss}_B = 0$, which is due to the fact that ReqLoss_B cannot be a function of h_A in any other case (since $y_A = 1$, there cannot be a rule $r \in \mathcal{R}^-_A$ such that $\text{ReqL}^p_B = 1 - \text{ReqL}_A$).

2. Suppose $y_B = 0$. Then, equivalently,

$$\text{ReqLoss}_B = -\log(1 - \text{ReqLoss}^n_B) \quad \text{and} \quad \frac{\partial \text{ReqLoss}_B}{\partial h_A} \leq 0.$$

This is due to the fact that:

- either $\text{ReqLoss}_B = h_A$, which is possible only if $A = B$ or there exists a rule $r \in \mathcal{R}^-_B$ such that $A \in body(r)$ whose body is satisfied by the ground truth and:

$$\max_{r \in \mathcal{R}^-_B}(\min_{l \in body(r)}(y_l \text{ReqL}_l)) = h_A.$$

Then,

$$\frac{\partial \text{ReqLoss}_B}{\partial h_A} = -\frac{1}{h_A} \leq 0,$$

- or $\text{ReqLoss}_B = 0$, which is due to the fact that ReqLoss_B cannot be a function of h_A in any other case (since $y_A = 1$, there cannot be a rule $r \in \mathcal{R}^+_A$ such that $\text{ReqL}^n_B = \text{ReqL}_A$). $\square$

4. Optimizing ReqL and ReqLoss for GPU

Neural networks are normally trained on GPUs, which allow for the execution of multiple operations in parallel. In this section, we show how to relax the assumption that each level has only one associated label, thus allowing for the exploitation of the GPUs' ability to execute multiple parallel operations to compute the value of ReqL, and hence ReqLoss, for all the labels at the same level.

Intuitively, given an MC problem with requirements $(\mathcal{P}, \Pi)$ and a label ordering λ, we can simultaneously compute the final value for all those labels that are not dependent one from the other.

Example 8. Consider $\Pi = \{\neg A_1 \vee A_2, \neg A_3 \vee A_4\}$, and assume that λ is a label ordering satisfying $\lambda(A_1) < \lambda(A_2)$ and $\lambda(A_3) < \lambda(A_4)$. In this case, we can easily see that, if we first simultaneously compute ReqL_{A_1} and ReqL_{A_3}, and then we compute simultaneously ReqL_{A_2} and ReqL_{A_4}, we get the same results as following the sequential order given by λ.

The notion of (in)dependency is based on the *dependency graph* associated with the set of rule $\mathcal{R} = \cup_{A \in \mathcal{A}}(\mathcal{R}^+_A \cup \mathcal{R}^-_A)$ computed from Π on the basis of λ, and defined as the directed acyclic graph (DAG) G

1. whose set of nodes is the set of labels $\mathcal{A}$, and
2. with an edge directed to $head(r)$ from each label occurring in $body(r)$, for some rule $r \in \mathcal{R}$.

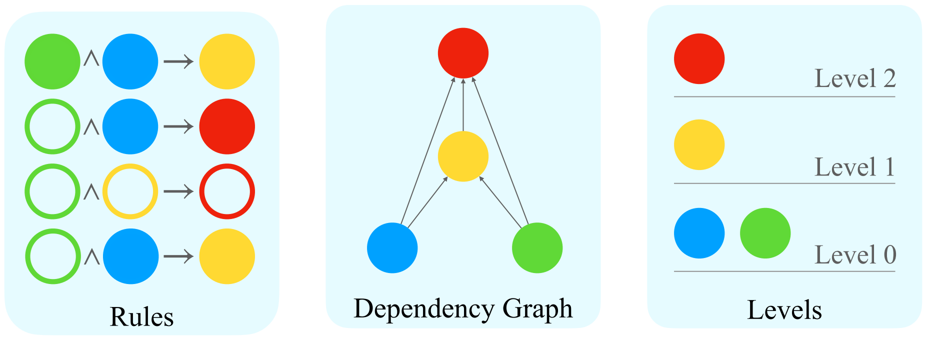


Fig. 9. Graphical representation of how levels are assigned.

Given the DAG G associated with $\mathcal{R}$, the *partial label ordering* assigns to each label A the level corresponding to its height in G, which is equal to the length of the longest path starting from a source node and ending in A. The source nodes are those having no incoming edges.

A visual representation of how the levels are assigned is given in Fig. 9. Given Π and λ, the partial label ordering λ' assigns to a label in a node the smallest possible value while retaining the order with the labels on which it depends. This is due to the fact that:

1. all the source nodes have level 0,
2. a label A has the level $i > 0$ if and only if there exists an edge from A_1 to A where A_1 has level $i - 1$ and all the edges to A start from a label with level $k \leq i - 1$.

Thanks to λ', we can compute ReqL and ReqLoss for the labels with the same level in parallel, thus exploiting GPUs' abilities. A visual overview of the steps necessary to obtain ReqL is given in Fig. 10. We refer to Section 4 in [5] for the details on how to transform the per label operations defined so far to matrix operations that are supported by GPUs and are able to compute the final value associated with multiple labels in parallel.

5. Experimental analysis

In this section, we show (i) how the proposed methodology can be effectively applied on challenging real-world problems, and (ii) how CCN$^+$ compares with respect to other state-of-the-art systems for MC problems with propositional logic requirements. We thus first present the experimental results of CCN$^+$ on ROAD-R, a very large safety-critical dataset for autonomous driving with propositional logic requirements [15]. On ROAD-R, we study the impact that the label ordering has on the performance of CCN$^+$, and we show that our layer and loss allow to improve the performance of different models for action detection in autonomous driving domains. Then, we consider 18 other real-world datasets and we compare CCN$^+$ with (i) Semantic Probabilistic Layer (SPL) [14], i.e., the current state-of-the-art model able to deal with, and also guaranteed to be compliant with, requirements expressed in full propositional logic, and (ii) 4 state-of-the-art models for MC problems whose predictions are made compliant with the requirements via post-processing.[3]

5.1. The ROAD-R dataset

The ROAD-R[4] dataset consists of:

1. 22 long ($\sim$ 8 minutes each) videos annotated with road events. A road event corresponds to a tube, i.e., a sequence of frame-wise bounding boxes linked in time. Each bounding box is labeled with a subset of the 41 labels specified in Table 1, which specify (i) the agent in the bounding box, (ii) the action(s) being made by the agent, and (iii) the location(s) of the agent, and
2. 243 propositional logic requirements expressing which are the admissible sets of labels that can be predicted for each bounding box. An example of a requirement is thus ($\neg$RedTL $\vee$ $\neg$GreenTL), expressing that a traffic light cannot be red and green at the same time.

In Fig. 11, we give an example of a prediction that is compliant with all requirements and one that violates the requirement ($\neg$RedTL $\vee$ $\neg$GreenTL).

As ROAD-R is an action detection dataset, we measure the tested models' performance using the *frame mean average precision* (f-mAP), which is the standard metric used in the field (see, e.g., [20,21]), with IoU threshold equal to 0.5, as in [15].

[3] Link to code: https://github.com/atatomir/CCN.

[4] Link to dataset: https://github.com/EGiunchiglia/ROAD-R.

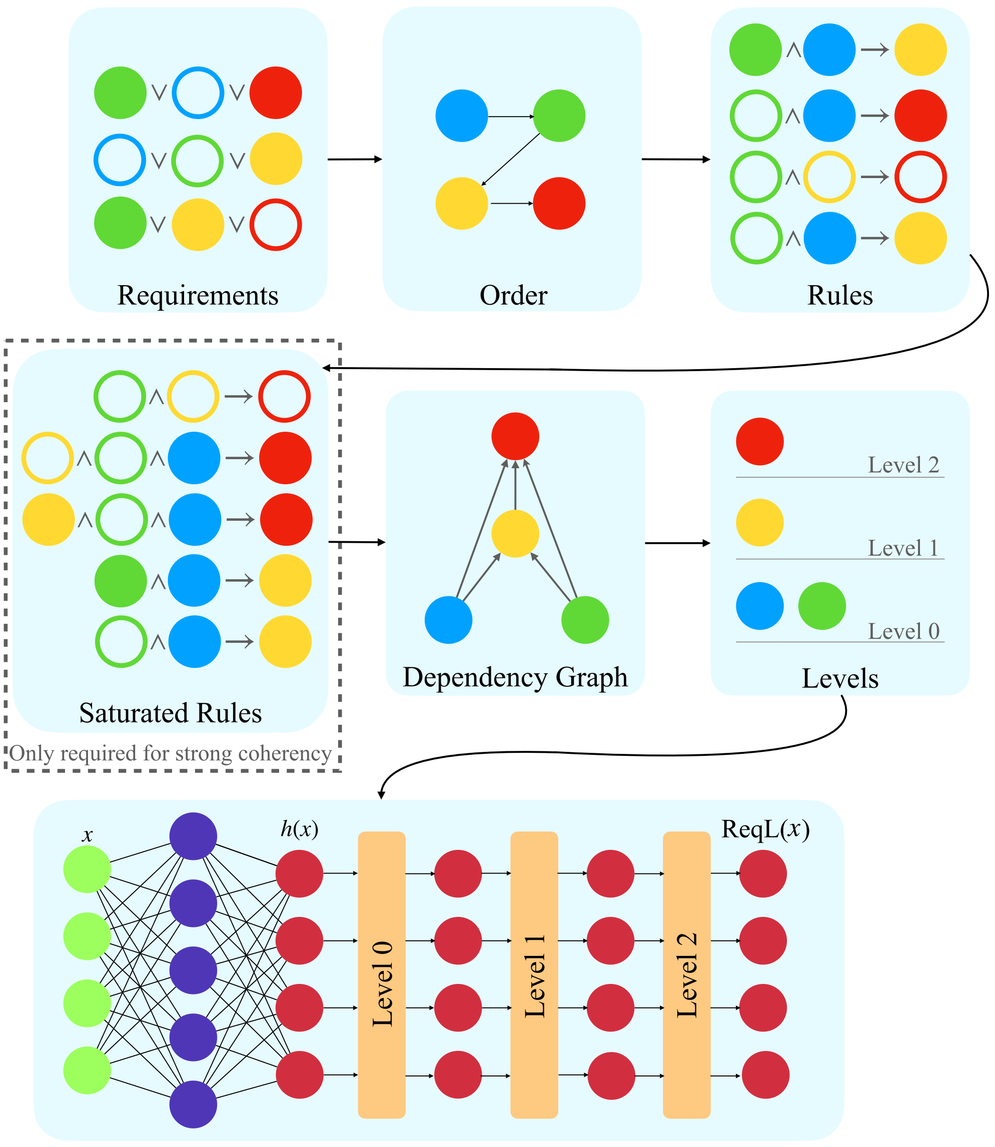


Fig. 10. Visual overview of how ReqL is created. Given a set of requirements, we can define a label ordering, which defines how the requirements are written as rules. If strong coherence is required, the rules are then saturated. From the rules, we then build the dependency graph, which in turn allows us to assign multiple labels to the same levels. Once we have the levels, we can finally build ReqL on top of any neural network h.

For all the considered models, we used a 3D-RetinaNet detector with a 2D-ConvNet backbone based on ResNet50 [22]. In all experiments, we trained each model for 30 epochs with an SGD optimizer [23], a batch size of 4, a sequence length of 8, an image input size of 512×682. All models used an initial learning rate of 0.0041, except for SlowFast, which used a value of 0.007, and the learning rate dropped by a factor of 10 after 20 and 25 epochs.

5.1.1. *Impact of the label ordering*

It is known that different variable orderings have an impact on the time and space needed to build ReqL, and that ReqL size can be exponential in the induced width of the constraints' interaction graph (see [19]). Despite this limitation, we never experienced memory problems in the ReqL construction for any variable ordering we considered, and the time needed to make a prediction has always been negligible.

To evaluate the effects of different variable orderings, we take one of the six state-of-the-art models from ROAD-R [15], we build ReqL on top of it, and we train the resulting model with ReqLoss. In particular, as a state-of-the-art model, we use the 3D-RetinaNet[5]

[5] https://github.com/gurkirt/3D-RetinaNets.

Table 1
Indexed labels in the ROAD-R Dataset.

Index	Agent	Index	Action	Index	Location
0	Pedestrian	10	Move away	29	AV lane
1	Car	11	Move towards	30	Outgoing lane
2	Cyclist	12	Move	31	Outgoing cycle lane
3	Motorbike	13	Brake	32	Incoming lane
4	Medium vehicle	14	Stop	33	Incoming cycle lane
5	Large vehicle	15	Indicating left	34	Pavement
6	Bus	16	Indicating right	35	Left pavement
7	Emergency vehicle	17	Hazard lights on	36	Right pavement
8	AV traffic light	18	Turn left	37	Junction
9	Other traffic light	19	Turn right	38	Crossing location
		20	Overtake	39	Bus stop
		21	Wait to cross	40	Parking
		22	Cross road from left		
		23	Cross road from right		
		24	Crossing		
		25	Push object		
		26	Red traffic light		
		27	Amber traffic light		
		28	Green traffic light		

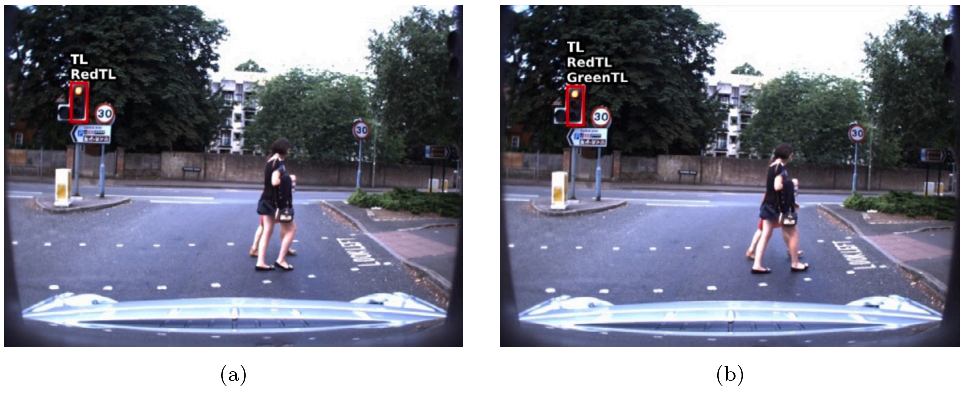


Fig. 11. (a) Example of a prediction that satisfies all requirements of ROAD-R. (b) Example of a prediction that violates the requirement ¬RedTL ∨ GreenTL.

object detector, as presented in [16], with an underlying Inflated 3D-ConvNet (I3D) [24] temporal feature learning model. In the remaining of the section, we call the I3D model trained without ReqL and ReqLoss the baseline model. Given the baseline model, we consider six different policies for choosing the label orderings:

1. *Random* (Rnd): we randomly shuffle the labels, and we take the resulting ordering.
2. *Frequency-based* (Freq): we order the labels from the ones that occur the most frequently to the ones that occur the least frequently in the ground truth associated with the data points used for training. This was done following the intuition that often it is easier to learn the distribution of labels for which we have a lot of positive examples.
3. *Reverse Frequency-based* (Rev-Freq): given the frequency-based ordering, we take the reverse.
4. *Simulation-based* (Sim): we order the labels from the ones for which the baseline model obtains the highest f-mAP on the validation set, to the ones that obtain the lowest f-mAP. This means that we order the labels from the easiest to learn, to the most difficult.
5. *Reverse Simulation-based* (Rev-Sim): given the simulation-based ordering, we take the reverse.

Indeed, the frequency and simulation-based orderings reflect the intuition to try to have at the bottom of the ordering the labels easier to predict. Their reverse counterparts and the random ordering have been included to evaluate the impact of the proposed orderings. We acknowledge that other variable orderings are possible, e.g., derived from works in propositional reasoning and/or knowledge compilation (see, e.g., [25,26]) and that such orderings may lead to more compact representation of the requirements and/or better performance. However, works in propositional reasoning and knowledge compilation concentrate on how to allow for efficient reasoning given the available knowledge expressed in propositional logic, while our focus is on how to get better performance at inference time given the available knowledge about the propositional logic requirements and the datapoints.

In Fig. 12, we plot the dependency graphs obtained with the different label orderings. As expected, we get a different number of levels depending on the ordering used, as we get: (i) 18 levels when using random ordering, (ii) 16 levels when using frequency-based ordering, (iii) 17 levels when using reverse frequency-based ordering, (iv) 19 levels when using simulation-based, and (v) 18 levels

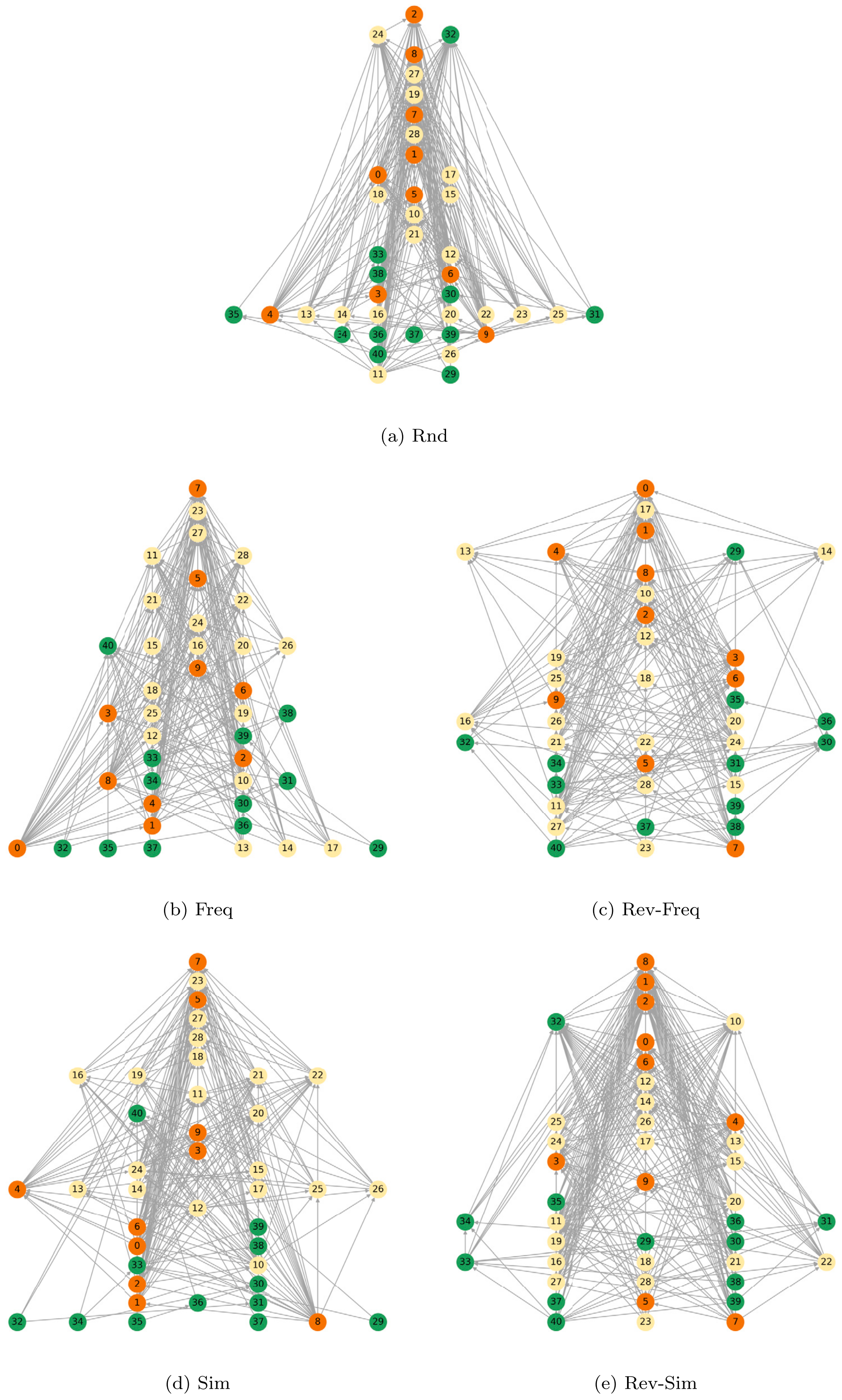
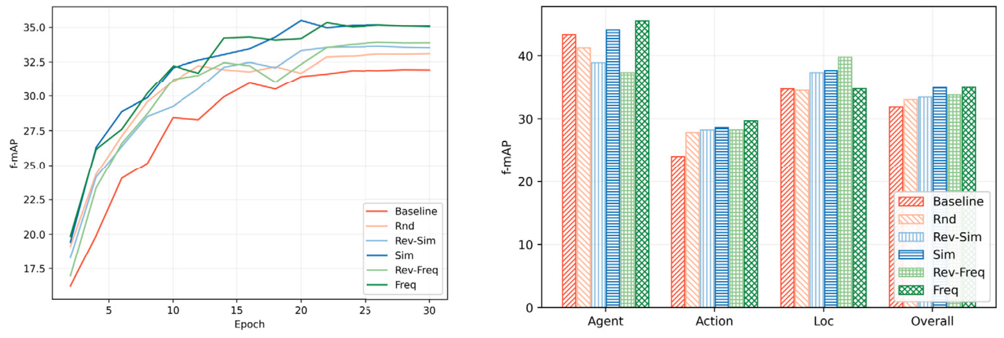


Fig. 12. Visualization of the dependency graphs obtained with different label orderings. Each number represents the index of each label as per Table 1. Orange, yellow, and green nodes represent agent, action, and location labels, respectively.



(a) f-mAP per epoch (b) Overall f-mAP and for each label type

Fig. 13. Performance in terms of f-mAP for the baseline model and CCN$^+$ when using different label orderings.

when using reverse simulation-based ordering. As we can see from the dependency graphs generated using the frequency-based and the simulation-based orderings, our intuition that frequent labels are easier to learn is correct. Indeed, e.g., we have that in both orderings the labels “Emergency Vehicle” and “Cross Road from Right” are computed last and second to last in both orderings. In the same way, 4 of the 6 labels appearing at the first level of the simulation-based dependency graph appear also in the first level of the frequency-based one (the remaining two, i.e., 34 and 8, both appear at the 4th level).

In the following, we report the results on the validation set. We first show the trend followed by the overall f-mAP over different epochs during training in Fig. 13a. Then, in Fig. 13b, we plot the f-mAP obtained by each model at the last epoch broken down by label type (i.e., agent, action, and location), together with the overall f-mAP.

As we can see from the first plot, the baseline gets the lowest f-mAP throughout the entire training, thus proving that including ReqL and ReqLoss not only makes neural networks safe, since they satisfy the requirements, but also improves their performance by exploiting the background knowledge that they express. Our second plot shows that the label ordering also deeply impacts on which labels we get the improvement in performance.

Indeed, if we focus on the results obtained with the reverse frequency-based and the reverse simulation-based, we can see that we obtain on average worse performance on the agent labels, which is though compensated by an improvement in performance over the action and location labels.

At the same time, we can see that well-chosen label orderings yield an improvement in performance over all label types, as happens with the frequency-based and the simulation-based orderings. This confirms our hypothesis that the label ordering has a drastic effect on the final performance, and, in particular, that the best strategy is to infer first the labels that appear most frequently (for frequency ordering) or those that are more accurately predicted by the unconstrained model (for simulation ordering). In what follows, we thus conduct our experimental analysis using the frequency-based and the simulation-based orderings.

5.1.2. Comparison with standard action detection models

We now study whether CCN$^+$ can help improve the performance of different models. To this end, we integrate into the 3D-RetinaNet object detector six different state-of-the-art temporal feature learning architectures:

1. *2D-ConvNet* (C2D) [27]: a Resnet50-based architecture with an additional temporal dimension for learning features from videos. The extension from 2D to 3D is done by adding a pooling layer over time to combine the spatial features.
2. *Inflated 3D-ConvNet* (I3D) [24]: a sequential learning architecture extendable to any state-of-the-art image classification model (2D-ConvNet-based), able to learn continuous spatio-temporal features from the sequence of frames.
3. *Recurrent Convolutional Network* (RCN) [28]: a 3D-ConvNet model that relies on recurrence for learning the spatio-temporal features at each network level. During the feature extraction phase, RCNs exploit both 2D convolutions across the spatial domain and 1D convolutions across the temporal domain.
4. *Random Connectivity Long Short-Term Memory* (RCLSTM) [29]: an updated version of LSTM in which the neurons are connected in a stochastic manner, rather than fully connected. In our case, the LSTM cell is used as a bottleneck in Resnet50 for learning the features sequentially.
5. *Random Connectivity Gated Recurrent Unit* (RCGRU) [29]: an alternative version of RCLSTM where the GRU cell is used instead of the LSTM one. GRU makes the process more efficient with fewer parameters than the LSTM.
6. *SlowFast* [30]: a 3D-CNN architecture that contains both slow and fast pathways for extracting the sequential features. A Slow pathway computes the spatial semantics at low frame rate, while a Fast pathway processes high frame rate for capturing the motion features. Both of the pathways are fused in a single architecture by lateral connections.

For each of the above models, (i) we train the model normally, (ii) we build CCN$^+$ on top of the model following the frequency-based ordering, and (iii) we build CCN$^+$ on top of the model following the simulation-based ordering. Furthermore, we saturate the set of requirements, and we train CCN$^+$ again following both the frequency-based ordering and the simulation-based ordering. When saturating the requirements, the number of levels does not change, while the numbers of obtained rules are (i) 273 rules with

Table 2

Performance in terms of f-mAP of the unconstrained state-of-the-art models and CCN$^+$ with (i) frequency-based and simulation-based label orderings and (ii) saturated and unsaturated requirements. The asterisk (*) next to the average ranking means that the difference in performance between that configuration of CCN$^+$ and the state-of-the-art models is statistically significant according to the Wilcoxon test.

		Π		Saturated Π	
	Baseline	CCN$^+$-Freq	CCN$^+$-Sim	CCN$^+$-Freq	CCN$^+$-Sim
I3D	29.30	30.37	**30.98**	29.89	30.36
C2D	26.34	**27.93**	26.57	27.92	26.27
RCN	29.26	**30.02**	29.32	29.50	29.16
RCGRU	29.24	**30.50**	30.39	30.14	29.93
RCLSTM	28.93	29.91	29.88	29.55	**30.42**
SlowFast	29.73	**31.88**	31.56	31.33	31.45
Avg.ranking	4.7	1.3*	2.3*	3.2*	3.5

Table 3

Summary of the real-world MC datasets with propositional logic requirements. For each dataset, we report from left to right: (i) name, (ii) number of features (D), (iii) number of classes (L), (iv-vi) number of data points for each split, (vii) number of constraints (C), (viii) number of different classes that appear at least once as head of a constraint (H), (ix) average number of classes appearing in the body (B), (x-xi) average number of classes appearing positively (resp., negatively) in the body (B^+ (resp., (B^-)), and (xii) percentage of classes appearing at least once as head of a constraint.

Dataset	D	L	Train	Val	Test	C	H	B	B^+	B^-	H/L
Arts	462	26	2975	525	1500	344	11	7.1	5.6	1.5	42.3%
Bibtex	1836	159	4148	732	2515	399	32	4.2	4.2	0.0	20.1%
Business	438	30	2975	525	1500	77	7	2.5	2.3	0.2	23.3%
Cal500	68	174	298	53	151	39	7	2.0	1.1	0.9	4.0%
Delicious	500	983	10982	1938	3185	104	61	1.0	1.0	0.0	6.1%
Emotions	72	6	332	59	202	1	1	5.0	0.0	5.0	16.7%
Enron	1001	53	954	169	579	7	5	1.0	1.0	0.0	9.4%
Genbase	1186	27	393	70	199	88	13	2.2	2.0	0.2	48.1%
Image	294	5	1190	210	600	1	1	4.0	0.0	4.0	20.0%
Medical	1449	45	283	50	645	17	9	1.2	1.2	0.0	20.0%
Rcv1subset1	944	101	3570	630	1800	247	16	3.7	2.9	0.8	15.8%
Rcv1subset2	944	101	3570	630	1800	81	15	2.4	1.7	0.7	14.9%
Rcv1subset3	944	101	3570	630	1800	72	16	2.3	1.7	0.6	15.8%
Rcv1subset4	944	101	3570	630	1800	63	14	2.1	1.7	0.4	13.9%
Rcv1subset5	944	101	3570	630	1800	73	11	2.5	2	0.5	10.9%
Science	743	40	2975	525	1500	37	11	2.1	1.7	0.4	27.5%
Scene	294	6	1029	182	1196	1	1	5.0	0.0	5.0	16.7%
Yeast	103	14	1275	225	917	34	11	2.3	1.9	0.4	78.6%

random ordering, (ii) 278 rules with frequency-based ordering, (iii) 266 rules with reverse frequency-based ordering, (iv) 304 rules with simulation-based ordering, and (v) 276 with reverse simulation-based ordering.

We report the results obtained with each model in Table 2. In the table, we first notice that (no matter the ordering and whether Π is saturated or not) CCN$^+$ achieves a better performance than its standard counterpart in all cases but two (i.e., for RCN and C2D with saturated Π and simulation-based ordering). Considering that we are using the same models with just an additional layer on top, such improvements are remarkable, as CCN$^+$ improves over the performance of its standard neural counterpart by up to 6.03% (for C2D) and 7.23% (for SlowFast). Secondly, we notice that the results obtained with the saturated requirements are generally lower than the ones obtained with their non-saturated counterpart (in 11 out of 12 comparisons the non-saturated versions perform better). This is reflected in their average rankings, which are equal to 1.3 and 2.3 when relying on the non-saturated requirements and equal to 3.2 and 3.5 when relying on the saturated ones. Finally, perhaps surprisingly, we notice that often the frequency-based ordering performs better than the simulation-based one. This is though a positive fact, as it means that there is no need to first train a baseline model to decide the label ordering, and it is enough to count the frequency of each label.

We also verified the statistical significance of the results following [31]. We thus performed the Wilcoxon test to compare the results obtained with the baseline model against all other models. We obtained the p-value < 0.5 for all tests but one, i.e., when we compare the baseline with CCN$^+$ with saturated Π and simulation-based ordering.

5.2. Multi-label classification datasets

To compare CCN$^+$ with other state-of-the-art models on a broad range of different scenarios, we considered the 18 datasets proposed in [5]. The list of the datasets together with a summary of their characteristics is reported in Table 3. The various datasets come from different application domains, in particular:

1. CAL500 and EMOTIONS are 2 music classification datasets [32,33],
2. GENBASE and YEAST are 2 functional genomics datasets [34,35],
3. IMAGE and SCENE are 2 image classification datasets [36,37], and
4. the remaining 10 are text classification datasets [38–41].[6]

Furthermore, as it can be seen from Table 3, they differ significantly both in the number of data points/classes (columns D and L) and in the characteristics of the associated sets of constraints. Indeed, we have datasets having just a few (one)/many constraints (column C), involving a few/many classes in the head (column H/L) and in the body, either positively or negatively (columns B, B^+, and B^-). Finally, for each dataset, we know that there exists a set of stratified rules that is equivalent to the set of constraints, thus allowing us to directly compare CCN$^+$ and CCN(h).

Regarding the set of metrics used in this experimental analysis, we follow the guidelines given in [42], and out of the 19 metrics that can be used in MC problems, we use:

1. average precision,
2. coverage error,
3. Hamming loss,
4. multi-label accuracy,
5. one-error,
6. ranking loss.

As the field of learning with requirements is very much still in its infancy, there are only three other models that have been proposed so far that are able to learn from the requirements while guaranteeing their satisfaction and these are: MultiplexNet [6], Semantic Probabilistic Layers (SPL) [14] and CCN(h) [5]. As the code for MultiplexNet is not publicly available, in our experiments, we compared CCN$^+$ against CCN(h) and SPL. Further, in order to show the importance of incorporating the requirements during training time, we also compared our model against four MC models whose predictions are made compliant with the requirements through an additional post-processing step. The considered models can be characterized by the order of classes correlations they exploit:

1. BR [37], a first-order model which considers each class separately, ignoring class correlations, and
2. ECC [43], RAKEL [44], and CAMEL [45], which exploit correlations among two or more classes.

BR, ECC, and RAKEL are well-established MC models, and CAMEL is the current state-of-the-art MC model [45].

Regarding the experimental analysis settings, all experiments were run on an Nvidia Titan RTX with 24 GB memory, and we applied the same preprocessing to all the datasets. All the categorical features were transformed using one-hot encoding. The missing values were replaced by their mean in the case of numeric features and by a vector of all zeros in the case of categorical ones. All the features were standardized. ECC, BR, and RAKEL were implemented using scikit-multilearn [46] by deploying the logistic regression model as the base classifier. On the other hand, we used the publicly available authors' implementation for CAMEL,[7] with the hyperparameters suggested by the authors. For all the models, we got results comparable to the ones reported in the work by [47]. Regarding SPL, we again used the publicly available authors' implementation.[8] However, for this model the optimal hyperparameters can highly vary. Thus, to ensure we find the optimal hyperparameters, instead of using the standard grid search, we used the more recently proposed Sequential Model-Based Optimization (SMBO) with Tree Parzen Estimator (TPE) [48].[9] The hyperparameters' search space used and the final hyperparameters found for each dataset can be found in Appendix A. On the contrary, CCN$^+$ was run using the same hyperparameters for the underlying neural network as CCN(h). The only search that we did was thus over the different label orderings. Learning from the experiments conducted on ROAD-R, we considered only four possible orderings:

1. *Balance-based*: we order the labels from the most balanced ones to the least.
2. *Reverse Balance-based*: given the balance-based ordering, we take the reverse.
3. *Simulation-based*: we order the labels from the ones for which the baseline model obtains the lowest binary cross-entropy loss on the validation set, to the ones that obtain the highest loss.
4. *Reverse Simulation-based*: given the simulation based ordering, we take the reverse.

We trained each model with all four possible orderings and we then reported the results for the one that achieved the lowest binary cross-entropy loss over the validation set. We chose to test CCN$^+$ without an extensive hyperparameters search to show that simply by using the same hyperparameters as CCN(h) and choosing a suitable label ordering: (i) we can perform as well as or better than CCN(h), and (ii) we can outperform all the other systems.

[6] Link to datasets: https://github.com/EGiunchiglia/CCN/tree/master/data/.

[7] Link: https://github.com/hinanmu/CAMEL.

[8] Link: https://github.com/KareemYousrii/SPL.

[9] Link to code: https://github.com/hyperopt/hyperopt.

Table 4
Comparison of CCN$^+$ with other state-of-the-art models. Best results are in bold.

	Model	Arts	Bibtex	Business	Cal500	Delicious	Emotions	Enron	Genbase	Image
Average Precision	CCN$^+$	0.619	0.584	0.894	**0.521**	0.345	**0.807**	**0.709**	0.997	**0.812**
	CCN(h)	0.623	0.586	**0.904**	0.520	0.347	0.800	0.704	0.996	0.807
	SPL	0.382	-	0.844	0.378	-	0.739	0.514	0.975	0.761
	CAMEL	**0.625**	**0.605**	0.899	0.513	**0.398**	0.756	0.708	0.990	0.793
	ECC	0.544	0.302	0.867	0.401	0.112	0.772	0.643	**1.000**	0.738
	BR	0.546	0.308	0.863	0.441	0.140	0.793	0.643	**1.000**	0.726
	RAKEL	0.530	0.317	0.856	0.433	0.140	0.798	0.636	**1.000**	0.721
Coverage Error	CCN$^+$	0.175	**0.104**	0.077	**0.734**	**0.519**	**0.314**	**0.213**	0.016	**0.185**
	CCN(h)	**0.172**	0.105	**0.065**	**0.734**	0.520	0.315	0.217	0.016	0.187
	SPL	0.485	-	0.126	0.872	-	0.389	0.448	0.032	0.267
	CAMEL	0.202	0.157	0.083	0.791	0.613	0.372	0.256	0.010	0.201
	ECC	0.223	0.801	0.089	0.853	0.987	0.338	0.285	**0.009**	0.242
	BR	0.217	0.802	0.086	0.789	0.989	0.324	0.288	**0.009**	0.245
	RAKEL	0.221	0.796	0.085	0.791	0.988	0.317	0.294	**0.009**	0.250
Hamming Loss	CCN$^+$	**0.054**	**0.012**	0.027	**0.135**	**0.018**	0.199	**0.046**	**0.001**	**0.166**
	CCN(h)	**0.054**	0.013	**0.023**	0.136	**0.018**	**0.197**	**0.046**	**0.001**	0.172
	SPL	0.059	-	0.027	0.158	-	0.233	0.047	0.005	0.170
	CAMEL	0.055	0.013	**0.023**	0.138	**0.018**	0.265	0.047	0.003	0.174
	ECC	0.081	0.013	0.031	0.172	0.019	0.245	0.055	**0.001**	0.218
	BR	0.079	0.013	0.032	0.162	**0.018**	0.229	0.054	**0.001**	0.232
	RAKEL	0.082	0.013	0.034	0.165	0.019	0.223	0.055	**0.001**	0.225
Multi-label Acc.	CCN$^+$	0.217	0.270	0.583	0.203	0.097	0.527	**0.395**	0.986	**0.499**
	CCN(h)	**0.238**	**0.272**	0.601	0.203	0.095	**0.534**	**0.395**	0.986	0.488
	SPL	0.144	-	0.605	0.203	-	0.479	0.383	0.915	0.463
	CAMEL	0.218	0.193	**0.609**	0.210	0.147	0.354	0.381	0.943	0.456
	ECC	0.217	0.250	0.548	0.220	0.114	0.446	0.361	**0.992**	0.387
	BR	0.217	0.260	0.538	0.221	0.150	0.465	0.365	**0.992**	0.369
	RAKEL	0.215	0.263	0.527	**0.222**	**0.152**	0.485	0.361	**0.992**	0.376
One-error	CCN$^+$	0.480	0.376	0.102	**0.113**	0.361	**0.253**	0.237	**0.000**	**0.289**
	CCN(h)	0.475	0.376	0.093	**0.113**	0.357	0.273	0.235	**0.000**	0.2936
	SPL	0.639	-	0.161	0.461	-	0.365	0.377	0.028	0.299
	CAMEL	**0.460**	**0.349**	**0.090**	0.133	**0.315**	0.381	**0.223**	0.020	0.310
	ECC	0.568	0.535	0.137	0.378	0.692	0.332	0.309	**0.000**	0.392
	BR	0.567	0.517	0.147	0.232	0.546	0.292	0.299	**0.000**	0.425
	RAKEL	0.586	0.513	0.159	0.232	0.567	0.292	0.304	**0.000**	0.430
Ranking Loss	CCN$^+$	0.118	**0.057**	0.037	**0.173**	**0.110**	**0.158**	**0.074**	0.002	**0.156**
	CCN(h)	**0.115**	0.058	**0.030**	**0.173**	**0.110**	0.161	0.076	0.003	0.159
	SPL	0.409	-	0.066	0.296	-	0.252	0.226	0.014	0.244
	CAMEL	0.136	0.078	0.040	0.189	0.117	0.237	0.086	**0.001**	0.177
	ECC	0.158	0.680	0.046	0.257	0.861	0.193	0.107	**0.001**	0.231
	BR	0.155	0.670	0.045	0.218	0.820	0.177	0.108	**0.001**	0.234
	RAKEL	0.159	0.659	0.044	0.220	0.815	0.169	0.112	**0.001**	0.242

For each dataset, we run the models 10 times, and the average for each of the metrics is reported in Tables 4 and 5, while in Table 6 we report the average rankings of each model. Firstly, if we focus on SPL we can see that we have dashes on 4 datasets out of 18 in Tables 4 and 5, which means that we did not manage to train the models for those datasets. In particular, in the case of Bibtex and Delicious we could not run the model as the GPU ran out of memory. Indeed, it would have required at least 30GB of GPU RAM to run the model on Bibtex and 80GB on Delicious. On the other hand, in the case of Rcv1Subset4 and Rcv1Subset5 we got the loss equal to NaN with all the tested hyperparameters combinations. When computing the average rankings, SPL was ranked last in those datasets. Focusing on Table 6, we can see that CCN$^+$ and CCN(h) have very similar performance, with CCN$^+$ having the best ranking on 4 metrics out of 6. This is expected, as both the models rely on the same mechanism and number of neurons. Among the other models, we see that CAMEL has the best average ranking on all metrics but multi-label accuracy where ECC outranks it. Overall, these results once again show the ability of CCN$^+$ to make predictions that are guaranteed to be compliant with a set of requirements while exploiting the background knowledge they express to achieve outstanding performance.

6. Related work

In this paper, we have shown how it is possible to create deep learning models that are coherent-by-construction with respect to a set of requirements over the output space expressed in propositional logic.

Traditionally, in the neuro-symbolic field, researchers have not focused on logical requirements that must be satisfied, but rather on soft logical constraints that can be violated. Such constraints are often used as a way to express background knowledge about the

Table 5
Comparison of CCN$^+$ with the other state-of-the-art models. The best results are in bold.

	Model	Medical	Rcv1S1	Rcv1S2	Rcv1S3	Rcv1S4	Rcv1S5	Science	Scene	Yeast
Average Precision	CCN$^+$	**0.867**	**0.642**	0.663	0.642	**0.678**	**0.579**	0.604	**0.872**	0.766
	CCN(h)	0.866	**0.642**	**0.666**	**0.647**	0.675	0.560	0.603	0.868	**0.768**
	SPL	0.331	0.414	0.471	0.478	-	-	0.476	0.784	0.722
	CAMEL	0.807	0.622	0.647	0.636	0.654	0.564	**0.614**	0.824	0.766
	ECC	0.823	0.549	0.575	0.585	0.609	0.529	0.502	0.794	0.724
	BR	0.823	0.536	0.563	0.572	0.600	0.524	0.500	0.781	0.743
	RAKEL	0.811	0.532	0.556	0.562	0.589	0.508	0.493	0.794	0.732
Coverage Error	CCN$^+$	0.036	**0.092**	0.091	**0.100**	0.093	0.112	0.136	**0.076**	**0.449**
	CCN(h)	**0.035**	**0.092**	**0.089**	0.103	**0.080**	**0.107**	**0.131**	0.077	0.452
	SPL	0.439	0.465	0.402	0.386	-	-	0.345	0.198	0.479
	CAMEL	0.036	0.131	0.115	0.123	0.103	0.130	0.162	0.106	0.457
	ECC	0.045	0.185	0.166	0.167	0.169	0.196	0.225	0.127	0.495
	BR	0.045	0.194	0.181	0.178	0.184	0.210	0.227	0.128	0.476
	RAKEL	0.049	0.201	0.180	0.185	0.195	0.209	0.225	0.123	0.481
Hamming Loss	CCN$^+$	0.014	**0.026**	**0.022**	**0.024**	0.020	0.026	**0.031**	**0.086**	**0.196**
	CCN(h)	**0.013**	**0.026**	**0.022**	**0.024**	**0.019**	**0.025**	**0.031**	0.092	**0.196**
	SPL	0.028	0.028	0.024	**0.024**	-	-	0.035	0.103	0.215
	CAMEL	0.024	0.027	**0.022**	**0.024**	0.021	**0.025**	**0.031**	0.109	**0.196**
	ECC	0.019	0.031	0.027	0.028	0.026	0.030	0.049	0.131	0.221
	BR	0.019	0.032	0.028	0.029	0.027	0.031	0.051	0.151	0.214
	RAKEL	0.019	0.033	0.029	0.030	0.027	0.031	0.051	0.130	0.225
Multi-label Acc.	CCN$^+$	**0.593**	**0.299**	**0.315**	0.283	0.321	**0.275**	0.271	**0.614**	**0.480**
	CCN(h)	0.589	0.296	0.310	**0.303**	**0.324**	**0.275**	0.255	0.607	**0.480**
	SPL	0.175	0.267	0.296	0.230	-	-	**0.272**	0.544	0.458
	CAMEL	0.284	0.204	0.222	0.210	0.257	0.223	0.217	0.528	**0.480**
	ECC	0.481	0.264	0.277	0.273	0.297	0.269	0.209	0.478	0.443
	BR	0.477	0.263	0.279	0.275	0.289	0.263	0.200	0.438	0.456
	RAKEL	0.481	0.263	0.272	0.268	0.290	0.258	0.201	0.481	0.445
One-error	CCN$^+$	**0.181**	**0.407**	0.401	0.414	**0.364**	**0.400**	0.491	**0.216**	0.234
	CCN(h)	**0.181**	0.413	**0.389**	**0.405**	0.379	0.402	0.494	0.224	0.234
	SPL	0.698	0.535	0.471	0.462	-	-	0.542	0.267	0.263
	CAMEL	0.285	0.413	0.397	0.413	0.399	0.414	**0.472**	0.287	**0.231**
	ECC	0.251	0.477	0.462	0.453	0.436	0.451	0.603	0.319	0.300
	BR	0.251	0.492	0.474	0.466	0.435	0.466	0.605	0.358	0.259
	RAKEL	0.266	0.488	0.481	0.471	0.439	0.474	0.606	0.329	0.270
Ranking Loss	CCN$^+$	0.025	**0.036**	0.036	**0.041**	0.040	0.048	0.099	**0.072**	**0.170**
	CCN(h)	**0.024**	**0.036**	**0.035**	0.046	**0.035**	**0.046**	**0.094**	0.073	0.172
	SPL	0.388	0.308	0.296	0.279	-	-	0.284	0.205	0.218
	CAMEL	0.026	0.051	0.048	0.050	0.046	0.054	0.117	0.101	0.173
	ECC	0.033	0.086	0.078	0.078	0.086	0.093	0.176	0.103	0.208
	BR	0.032	0.091	0.088	0.085	0.097	0.101	0.177	0.131	0.190
	RAKEL	0.037	0.093	0.088	0.089	0.103	0.102	0.180	0.127	0.200

Table 6
Average ranking for each metric and model.

Metric	Average ranking						
	CCN$^+$	CCN(h)	SPL	CAMEL	ECC	BR	RAKEL
Average precision	**1.89**	2.08	6.78	2.86	4.53	4.64	5.22
Coverage Error	**1.72**	1.75	6.89	3.22	4.61	4.86	4.89
Hamming loss	**1.94**	1.97	4.81	2.83	5.11	4.67	5.28
Multi-label accuracy	2.42	**2.36**	4.83	4.78	4.50	4.58	4.53
One-error	2.06	**2.03**	6.00	2.75	4.86	4.81	5.50
Ranking loss	**1.69**	1.75	7.00	3.14	4.64	4.67	5.11

problem at hand, and they are incorporated into deep learning models' training to either alleviate their data greediness or to improve their performance. The most popular and intuitive way of incorporating such constraints in the training of deep learning models is to embed them in the loss function. The first proposed loss functions simply mapped the constraints to loss functions following the rules dictated by the t-norms [10,49–53]. Among the limitations that these losses present (for a thorough study on their limitations, see [54,55]), the most striking one is the fact that they are syntax-dependent, i.e., how a constraint is written impacts the final value associated with the loss function. To alleviate such problems, different alternatives were proposed to create syntax-independent loss functions (see, e.g., [8,9]). On the other hand, Li et al. [56] focus on the problem that the above loss-based approaches tend to only vacuously satisfy logical constraints through shortcuts, and thus they present a new framework that forces the model to avoid such shortcuts. Finally, Ahmed et al. [57] propose a way of modifying the established method of entropy regularization [58] to take into account logical constraints as well. They call this novel regularization method neuro-symbolic entropy regularization. An alternative to the incorporation of the constraints in the loss is proposed in [59,60], where the authors introduce an iterative method to embed structured information expressed by first-order logic (FOL) formulas into the weights of different kinds of deep neural networks. At each step, they consider a teacher network based on the set of FOL rules to train a student network to fit both supervisions and logic rules. On the other hand, in [61], the authors augment their models by assigning semantics via logical rules to their neurons. Indeed, some neurons are associated with logical predicates, and then their activation is modified depending on the activation of the neurons corresponding to predicates that co-occur in the same rules. Finally, DeepProblog [62] can incorporate logical constraints expressed in first-order logic, and to do so it uses Prolog's (later extended as Problog's [63]) backward chaining algorithm whose probabilistic weights are instead returned by a deep learning model. We refer to the survey [64] for an in-depth overview of deep learning with logical constraints, and to the survey [65] for a general overview of the neuro-symbolic field.

Recently, given the rise of applications of deep learning models in safety-critical scenarios, it has become more and more evident the need to be able to impose requirements (or hard logical constraints) on neural networks, as advocated in [7]. In this spirit, we can find the many works proposed by the hierarchical multi-label community (HMC) (see, e.g., [11,13,66,67]), where a prediction is considered valid only if it respects the hierarchy in which the labels are arranged. Despite the many interesting models proposed in this field, the requirements considered have a very limited syntax, as they all have the form $A \rightarrow B$, where A and B are labels of the considered problems. More recently, other works able to deal with more expressive constraints have been proposed in the neuro-symbolic field. The first one was CCN [5,68], which is the model from which we took inspiration in this paper. CCN is again made of two main components: (i) a layer that can be built on top of any neural network guaranteeing the satisfaction of the constraints, and (ii) a loss function ensuring that the gradients always flow in the right direction. However, the main limitation of CNN is given by the fact that it can only consider constraints expressed as normal logic rules, which are far less expressive than propositional logic. Later, MultiplexNet [6] was proposed. MultiplexNet considers more expressive constraints, as its constraints can consist of any quantifier-free linear arithmetic formula over the rationals (thus, involving "$+$", "$\geq$", "$\neg$", "$\wedge$", and "$\vee$"). MultiplexNet also builds an output layer on top of neural networks, however, in order to do so it needs to rewrite the constraints in disjunctive normal form, which though leads to an exponential explosion of the size of the constraints. Another model that is able to guarantee the satisfaction of the constraints is NESTER [69], which enforces the constraints not through a layer but through a constraint program taking as input the output of a neural network, thanks to which it can enforce both hard and soft constraints over both categorical and numerical variables. The most recent proposed model for imposing hard logical constraints is Semantic Probabilistic Layer [14], which is a layer that can be built on top of any neural network and guarantees the satisfaction of the constraints. SPL builds two circuits: a probabilistic circuit that has learnable parameters and a deterministic circuit encoding the background knowledge. In the worst case, the compiled circuit is exponential in the size of the constraints, which, in our experiments, does not represent an issue in most cases.

7. Conclusions

In this paper, we introduced CCN$^+$, a neuro-symbolic framework that, given a neural network and a set of requirements expressed in propositional logic, is able to (i) guarantee that the requirements are always satisfied, and (ii) exploit the background knowledge expressed by the requirements to improve the network's performance. We implemented CCN$^+$ in a GPU-friendly way, making its integration with neural networks seamless. Furthermore, we also provided an optimization in order to leverage GPU's architectures. Finally, we tested CCN$^+$ on 19 real-world multi-label classification datasets with propositional logic requirements (including a challenging dataset for autonomous driving), and showed that CCN$^+$ is able to outperform both its standard neural counterparts and the state-of-art-models. In the future, we plan to study how to determine the best label ordering for optimal performance and/or minimal layer size.

CRediT authorship contribution statement

Eleonora Giunchiglia: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, Supervision, Writing – original draft, Writing – review & editing. **Alex Tatomir:** Formal analysis, Methodology, Software, Visualization, Writing – original draft. **Mihaela Cătălina Stoian:** Software, Validation, Visualization, Writing – original draft. **Thomas Lukasiewicz:** Funding acquisition, Supervision, Validation, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data is publicly available (link is given in the paper). The code is available at: https://github.com/atatomir/CCN.

Acknowledgements

Mihaela Cătălina Stoian is supported by the EPSRC under the grant EP/T517811/1. This work was also supported by the AXA Research Fund and by the EU TAILOR grant 952215. We also acknowledge the use of the EPSRC-funded Tier 2 facility JADE (EP/P020275/1) and the GPU computing support by Scan Computers International Ltd.

Appendix A. SPL hyperparameters

The hyperparameters search space for SPL was defined in the following way:

1. for batch size we considered as possible values in $\{2, 4, 6, 8\}$,
2. for dropout ratio we considered the possible values in $\{0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$,
3. for hidden dimension we considered all possible hidden dimensions from 50 to 4000 with a step size of 50,
4. for the number of hidden layers we considered the possible values in $\{1, 2, 3, 4, 5\}$,
5. for the learning rate we considered all possible values in $[10^{-5}, 10^{-3}]$,
6. for the learning rate we considered all possible values in $[10^{-6}, 10^{-3}]$,
7. for the over-parametrization factor we considered the values in $\{1, 2, 3\}$,
8. for the number of mixtures we considered the values $\{1, 2, 3, 4\}$,
9. for the number of gating functions we considered the values $\{8, 16, 32, 64, 128, 256, 512, 1024\}$, and
10. for the number of gating layers we considered the values $\{1, 2, 3, 4\}$.

We let the hyperparameter search for each dataset run for 10 days or until 200 trials were completed (i.e., 200 different hyperparameters combinations were tested).

References

[1] A. Khashman, Face recognition using neural networks and pattern averaging, in: Proceedings of ISNN, vol. 3972, 2006.
[2] L. Yu, S. Wang, K.K. Lai, A neural-network-based nonlinear metamodeling approach to financial time series forecasting, Appl. Soft Comput. 9 (2) (2009).
[3] S. Zhang, L. Yao, A. Sun, Deep learning based recommender system: a survey and new perspectives, CoRR, arXiv:1707.07435 [abs], 2017.
[4] T. Shaikhina, N.A. Khovanova, Handling limited datasets with neural networks in medical applications: a small-data approach, Artif. Intell. Med. 75 (2017).
[5] E. Giunchiglia, T. Lukasiewicz, Multi-label classification neural networks with hard logical constraints, J. Artif. Intell. Res. 72 (2021).
[6] N. Hoernle, R. Karampatsis, V. Belle, K. Gal, MultiplexNet: towards fully satisfied logical constraints in neural networks, in: Proceedings of AAAI, 2022.
[7] E. Giunchiglia, F. Imrie, M. van der Schaar, T. Lukasiewicz, Machine learning with requirements: a Manifesto, arXiv:2304.03674, 2023.
[8] J. Xu, Z. Zhang, T. Friedman, Y. Liang, G. Van den Broeck, A semantic loss function for deep learning with symbolic knowledge, in: Proceedings of ICML, 2018.
[9] M. Fischer, M. Balunovic, D. Drachsler-Cohen, T. Gehr, C. Zhang, M. Vechev, DL2: training and querying neural networks with logic, in: Proceedings of ICML, 2019.
[10] M. Diligenti, M. Gori, M. Maggini, L. Rigutini, Bridging Logic and Kernel Machines, Machine Learning, 2012.
[11] J. Wehrmann, R. Cerri, R. Barros, Hierarchical multi-label classification networks, in: Proceedings of ICML, 2018.
[12] R. Cerri, R. Barros, A.C.P. de Leon Ferreira de Carvalho, Y. Jin, Reduction strategies for hierarchical multi-label classification in protein function prediction, BMC Bioinform. 17 (2016).
[13] E. Giunchiglia, T. Lukasiewicz, Coherent hierarchical multi-label classification networks, in: Proceedings of NeurIPS, 2020.
[14] K. Ahmed, S. Teso, K. Chang, G.V. den Broeck, A. Vergari, Semantic probabilistic layers for neuro-symbolic learning, in: Proceedings of NeurIPS, 2022.
[15] E. Giunchiglia, M.C. Stoian, S. Khan, F. Cuzzolin, T. Lukasiewicz, ROAD-R: the autonomous driving dataset for learning with requirements, Mach. Learn. (2023).
[16] G. Singh, S. Akrigg, M.D. Maio, V. Fontana, R.J. Alitappeh, S. Khan, S. Saha, K.J. Saravi, F. Yousefi, J. Culley, T. Nicholson, J. Omokeowa, S. Grazioso, A. Bradley, G.D. Gironimo, F. Cuzzolin, ROAD: the ROad event Awareness Dataset for autonomous driving, IEEE Trans. Pattern Anal. Mach. Intell. (2023).
[17] G. Metcalfe, Fundamentals of fuzzy logics, https://www.logic.at/tbilisi05/Metcalfe-notes.pdf, Sep 2005.
[18] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, in: Proceedings of ICLR, 2015.
[19] R. Dechter, I. Rish, Directional resolution: the Davis-Putnam procedure, revisited, in: Proceedings of KR, 1994.
[20] V. Kalogeiton, P. Weinzaepfel, V. Ferrari, C. Schmid, Action tubelet detector for spatio-temporal action localization, in: Proceedings of ICCV, 2017.
[21] D. Li, Z. Qiu, Q. Dai, T. Yao, T. Mei, Recurrent tubelet proposal and recognition networks for action detection, in: Proceedings of ECCV, 2018.
[22] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of CVPR, 2016.
[23] Y. LeCun, L. Bottou, G.B. Orr, K. Müller, Efficient backprop, in: Neural Networks: Tricks of the Trade, second edition, in: LNCS, vol. 7700, 2012.
[24] J. Carreira, A. Zisserman, Quo vadis, action recognition? A new model and the kinetics dataset, in: Proceedings of CVPR, 2017.
[25] A. Biere, M. Heule, H. van Maaren, T. Walsh (Eds.), Handbook of Satisfiability, second edition, Frontiers in Artificial Intelligence and Applications, vol. 336, IOS Press, 2021.
[26] A. Darwiche, P. Marquis, A knowledge compilation map, J. Artif. Intell. Res. (2002).
[27] X. Wang, R. Girshick, A. Gupta, K. He, Non-local neural networks, in: Proceedings of CVPR, 2018.
[28] G. Singh, F. Cuzzolin, Recurrent convolutions for causal 3D CNNs, in: Proceedings of ICCV Workshops, 2019.
[29] Y. Hua, Z. Zhao, Z. Liu, X. Chen, R. Li, H. Zhang, Traffic prediction based on random connectivity in deep learning with long short-term memory, in: Proceedings of VTC-Fall, 2018.
[30] C. Feichtenhofer, H. Fan, J. Malik, K. He, Slowfast networks for video recognition, in: Proceedings of ICCV, 2019.
[31] J. Demsar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006).
[32] D. Turnbull, L. Barrington, D. Torres, G. Lanckriet, Semantic annotation and retrieval of music and sound effects, IEEE Trans. Audio Speech Lang. Process. (2008).
[33] G. Tsoumakas, I. Katakis, I. Vlahavas, Effective and efficient multilabel classification in domains with large number of labels, in: Proceedings of ECML/PKDD—Workshop on Mining Multidimensional Data, 2008.
[34] S. Diplaris, G. Tsoumakas, P.A. Mitkas, I. Vlahavas, Protein classification with multiple algorithms, in: P. Bozanis, E.N. Houstis (Eds.), Advances in Informatics, 2005.
[35] A. Elisseeff, J. Weston, A kernel method for multi-labelled classification, in: Proceedings of NeurIPS, 2001, pp. 681–687.

[36] M.-L. Zhang, Z.-H. Zhou, ML-KNN: a lazy learning approach to multi-label learning, Pattern Recognit. (2007).
[37] M.R. Boutell, J. Luo, X. Shen, C.M. Brown, Learning multi-label scene classification, Pattern Recognit. 37 (9) (2004).
[38] J.P. Pestian, C. Brew, P. Matykiewicz, D.J. Hovermale, N. Johnson, K.B. Cohen, W. Duch, A shared task involving multi-label classification of clinical free text, in: Proceedings of Workshop on BioNLP, 2007.
[39] J. Read, B. Pfahringer, G. Holmes, Multi-label classification using ensembles of pruned sets, in: Proceedings of IEEE ICDM, 2008.
[40] A.N. Srivastava, B. Zane-Ulman, Discovering recurring anomalies in text reports regarding complex space systems, in: Proceedings of IEEE Aerospace Conference, 2005.
[41] G. Tsoumakas, I. Vlahavas, Random k-labelsets: an ensemble method for multilabel classification, in: J.N. Kok, J. Koronacki, R.L.d. Mantaras, S. Matwin, D. Mladenič, A. Skowron (Eds.), Proceedings of ECML, 2007.
[42] R.B. Pereira, A. Plastino, B. Zadrozny, L.H. Merschmann, Correlation analysis of performance measures for multi-label classification, Inf. Process. Manag. (2018).
[43] J. Read, B. Pfahringer, G. Holmes, E. Frank, Classifier chains for multi-label classification, in: W. Buntine, M. Grobelnik, D. Mladenić, J. Shawe-Taylor (Eds.), Machine Learning and Knowledge Discovery in Databases, 2009.
[44] G. Tsoumakas, A. Dimou, E. Spyromitros, V. Mezaris, I. Kompatsiaris, I. Vlahavas, Correlation-based pruning of stacked binary relevance models for multi-label learning, in: Proceedings of International Workshop on Learning from Multi-Label Data, 2009.
[45] S. Feng, P. Fu, W. Zheng, A hierarchical multi-label classification method based on neural networks for gene function prediction, Biotechnol. Biotechnol. Equip. 32 (2018).
[46] P. Szymanski, T. Kajdanowicz, A scikit-based python environment for performing multi-label classification, CoRR, arXiv:1702.01460 [abs], 2017, arXiv:1702.01460.
[47] L. Feng, B. An, S. He, Collaboration based multi-label learning, in: Proceedings of AAAI, 2019, pp. 3550–3557.
[48] J. Bergstra, D. Yamins, D.D. Cox, Making a science of model search: hyperparameter optimization in hundreds of dimensions for vision architectures, in: Proceedings of ICML, 2013.
[49] M. Diligenti, S. Roychowdhury, M. Gori, Integrating prior knowledge into deep learning, in: Proceedings of ICMLA, 2017.
[50] G. Marra, F. Giannini, M. Diligenti, M. Gori, LYRICS: a general interface layer to integrate logic inference and deep learning, in: Proceedings of ECML-PKDD, 2019.
[51] L. Serafini, A. d'Avila Garcez, Logic tensor networks: deep learning and logical reasoning from data and knowledge, in: Proceedings of NeSy-HLAI, 2016.
[52] S. Badreddine, A. d'Avila Garcez, L. Serafini, M. Spranger, Logic tensor networks, Artif. Intell. 303 (2022).
[53] M.C. Stoian, E. Giunchiglia, T. Lukasiewicz, Exploiting t-norms for deep learning in autonomous driving, in: Proceedings of the International Workshop on Neural-Symbolic Learning and Reasoning, 2023.
[54] E. van Krieken, E. Acar, F. van Harmelen, Analyzing differentiable fuzzy implications, in: Proceedings of KR, 2020.
[55] E. van Krieken, E. Acar, F. van Harmelen, Analyzing differentiable fuzzy logic operators, Artif. Intell. 302 (2022).
[56] Z. Li, Z. Liu, Y. Yao, J. Xu, T. Chen, X. Ma, J. Lü, Learning with logical constraints but without shortcut satisfaction, in: Proceedings of ICLR, 2023.
[57] K. Ahmed, E. Wang, K. Chang, G.V. den Broeck, Neuro-symbolic entropy regularization, in: Proceedings of UAI, 2022.
[58] Y. Grandvalet, Y. Bengio, Semi-supervised learning by entropy minimization, in: Proceedings of NeurIPS, 2004.
[59] Z. Hu, X. Ma, Z. Liu, E. Hovy, E. Xing, Harnessing deep neural networks with logic rules, in: Proceedings of ACL, 2016.
[60] Z. Hu, Z. Yang, R. Salakhutdinov, E. Xing, Deep neural networks with massive learned knowledge, in: Proceedings of EMNLP, 2016.
[61] T. Li, V. Srikumar, Augmenting neural networks with first-order logic, in: Proceedings of ACL, 2019.
[62] R. Manhaeve, S. Dumancic, A. Kimmig, T. Demeester, L. De Raedt, DeepProbLog: neural probabilistic logic programming, in: Proceedings of NeurIPS, 2018.
[63] L.D. Raedt, A. Kimmig, H. Toivonen, ProbLog: a probabilistic Prolog and its application in link discovery, in: Proceedings of IJCAI, 2007.
[64] E. Giunchiglia, M.C. Stoian, T. Lukasiewicz, Deep learning with logical constraints, in: Proceedings of IJCAI, 2022.
[65] A. d'Avila Garcez, M. Gori, L. Lamb, L. Serafini, M. Spranger, S. Tran, Neural-symbolic computing: an effective methodology for principled integration of machine learning and reasoning, FLAP 6 (2019).
[66] C. Vens, J. Struyf, L. Schietgat, S. Dzeroski, H. Blockeel, Decision trees for hierarchical multi-label classification, Mach. Learn. 73 (2008).
[67] C.N.J. Silla, A.A. Freitas, A survey of hierarchical classification across different application domains, Data Min. Knowl. Discov. 22 (2011).
[68] E. Giunchiglia, Deep learning with hard logical constraints, Ph.D. thesis, University of Oxford, 2022.
[69] P. Dragone, S. Teso, A. Passerini, Neuro-symbolic constraint programming for structured prediction, in: Proceedings of IJCLR-NeSy, 2021.