Machine Learning (2023) 112:3261–3291
https://doi.org/10.1007/s10994-023-06322-z

ROAD-R: the autonomous driving dataset with logical requirements

Eleonora Giunchiglia[1] · Mihaela Cătălina Stoian[1] · Salman Khan[2] · Fabio Cuzzolin[2] · Thomas Lukasiewicz[1,3]

Received: 7 June 2022 / Revised: 31 January 2023 / Accepted: 20 February 2023 /
Published online: 1 May 2023

Abstract
Neural networks have proven to be very powerful at computer vision tasks. However, they often exhibit unexpected behaviors, acting against background knowledge about the problem at hand. This calls for models (i) able to learn from requirements expressing such background knowledge, and (ii) guaranteed to be compliant with the requirements themselves. Unfortunately, the development of such models is hampered by the lack of real-world datasets equipped with formally specified requirements. In this paper, we introduce the ROad event Awareness Dataset with logical Requirements (ROAD-R), the first publicly available dataset for autonomous driving with requirements expressed as logical constraints. Given ROAD-R, we show that current state-of-the-art models often violate its logical constraints, and that it is possible to exploit them to create models that (i) have a better performance, and (ii) are guaranteed to be compliant with the requirements themselves.

Keywords Deep learning · Requirements · Logical constraints · Safety

1 Introduction

Neural networks have proven to be incredibly powerful at processing low-level inputs, and for this reason they have been extensively applied to computer vision tasks, such as image classification, object detection, and action detection. However, they can exhibit unexpected

Editors: AlirezaTamaddoni-Nezhad, Alan Bundy, Luc De Raedt, Artur d'Avila Garcez, Sebastijan Dumanić,Cèsar Ferri, Pascal Hitzler, Nikos Katzouris, Denis Mareschal, Stephen Muggleton, Ute Schmid

✉ Eleonora Giunchiglia
eleonora.giunchiglia@cs.ox.ac.uk

✉ Mihaela Cătălina Stoian
mihaela.stoian@cs.ox.ac.uk

[1] Department of Computer Science, University of Oxford, Oxford, UK

[2] School of Engineering, Computing and Mathematics, Oxford Brookes University, Oxford, UK

[3] Institute of Logic and Computation, TU Wien, Vienna, Austria

Table 1 ROAD labels

Agents:	Pedestrian, Car, Cyclist, Motorbike, Medium vehicle, Large vehicle, Bus, Emergency vehicle, Traffic light (TL), Other TL.
Actions:	Move away, Move towards, Move, Brake, Stop, Indicating left, Indicating right, Hazards lights on, Turn left, Turn right, Overtake, Wait to cross, Cross from left, Cross from right, Crossing, Push object, Red TL, Amber TL, Green TL.
Locations:	AV lane, Outgoing lane, Outgoing cycle lane, Incoming lane, Incoming cycle lane, Pavement, Left pavement, Right pavement, Junction, Crossing location, Bus stop, Parking.

behaviors, contradicting known requirements expressing background knowledge. This can have dramatic consequences, especially in safety-critical scenarios such as autonomous driving. To address the problem, models should (i) be able to learn from the requirements, and (ii) be guaranteed to be compliant with the requirements themselves. Indeed, as suggested in Amodei et al. (2016), in such settings it is of primary importance to create models that are able to operate within boundaries specified by the requirements written by domain experts. Unfortunately, the development of such models is hampered by the lack of real-world datasets equipped with formally specified requirements. A notable exception is given by hierarchical multi-label classification (HMC) problems (see, e.g., (Vens et al. 2008; Schietgat et al. 2010; Wehrmann et al. 2018)) in which datasets are provided with simple binary constraints of the form ($A \rightarrow B$) stating that label B must be predicted whenever label A is predicted.

In this paper, we generalize HMC problems by introducing *multi-label classification problems with (full) propositional logic requirements*. Thus, given a multi-label classification problem with labels A, B, and C, we can, for example, write the requirement:

$$(\neg A \wedge B) \vee C,$$

stating that for each data point in the dataset either the label C is predicted, or B but not A are predicted. Then, we present the ROad event Awareness Dataset with logical Requirements (ROAD-R), the first publicly available dataset for autonomous driving with requirements expressed as logical constraints. ROAD-R extends the ROAD dataset (Singh et al. 2022), which was built on top of the Oxford RobotCar Dataset (Maddern et al. 2017) and consists of 22 relatively long (~8 minutes each) videos annotated with *road events*. A road event corresponds to a tube/tubelet, 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. The goal is to predict the set of labels associated with each bounding box. We manually annotated ROAD-R with 243 constraints expressing which combinations of labels are admissible. We verified that the constraints hold for all bounding boxes' ground truth annotations appearing in the dataset using the SAT-solver MiniSat (Eén and Sörensoon 2004).[1]An example of a constraint is thus "a traffic light cannot be red and green at the same time", while there are no constraints like "pedestrians should cross at crossings", which should always be satisfied in theory, but which might not be in real-world scenarios.

Given ROAD-R, we considered 6 current state-of-the-art (SOTA) models, and we showed that they are not able to learn the requirements just from the data points, as more

[1] Link: http://minisat.set.

than 90% of their predictions violate the constraints. Then, we faced the problem of how to leverage the additional knowledge provided by constraints with the goal of (i) improving their performance, measured by the frame mean average precision (f-mAP) at intersection over union (IoU) thresholds 0.5 and 0.75; see, e.g., (Kalogeiton et al. 2017; Li et al. 2018), and (ii) guaranteeing that they are compliant with the constraints. To achieve the above two goals, we propose the following new models:

1. CL models, i.e., models with a *constrained loss* allowing them to learn from the requirements,
2. CO models, i.e, models with a *constrained output* enforcing the requirements on the output, and
3. CLCO models, i.e., models with both a constrained loss and a constrained output.

In particular, we consider three different ways to build CL (resp., CO, CLCO) models. More specifically, we run the 9×6 models obtained by equipping the 6 current SOTA models with a constrained loss and/or a constrained output, and we show that it is always possible to

1. Improve the performance of each SOTA model, and
2. Be compliant with (i.e., strictly satisfy) the constraints.

Overall, the best performing model (for IoU = 0.5 and also IoU = 0.75) is CLCO-RCGRU, i.e., the SOTA model RCGRU equipped with both constrained loss and constrained output: CLCO-RCGRU (i) always satisfies the requirements and (ii) has f-mAP = 31.81 for IoU = 0.5, and f-mAP = 17.27 for IoU = 0.75. On the other hand, the standard RCGRU model (i) produces predictions that violate the constraints at least 92% of the times, and (ii) has f-mAP = 30.78 for IoU = 0.5 and f-mAP = 15.98 for IoU = 0.75.

The main contributions of this paper are thus as follows:

1. We introduce multi-label classification problems with propositional logic requirements,
2. We introduce ROAD-R, which is the first publicly available dataset whose requirements are expressed in full propositional logic,
3. We consider 6 SOTA models and show that on ROAD-R, they produce predictions violating the requirements more than ~90% of the times,
4. We propose new models with a constrained loss and/or constrained output, and
5. We conduct an extensive experimental analysis and show that, with our new models, it is always possible to improve the performance of the SOTA models and satisfy the requirements.

The rest of this paper is organized as follows. After the introduction to the problem, we present ROAD-R (Sect. 3), followed by the evaluation of the SOTA models (Sect. 4) and of the SOTA models incorporating the requirements (Sect. 5) on ROAD-R. We end the paper with the related work (Sect. 6) and the summary and outlook (Sect. 7).

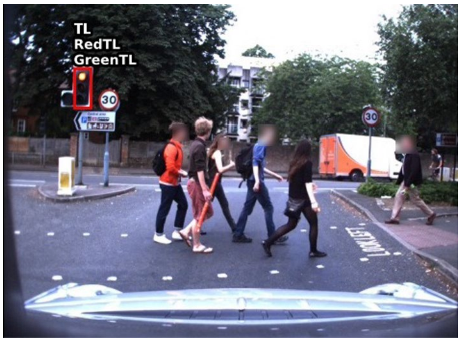


Fig. 1 Example of violation of $\neg$RedTL $\vee$ $\neg$GreenTL

2 Learning with requirements

In ROAD, the detection of road events requires the following tasks: (i) identify the bounding boxes, (ii) associate with each bounding box a set of labels, and (iii) form a tube from the identified bounding boxes with the same labels. Here, we focus on the second task, and we formulate it as a multilabel classification problem with requirements.

A *multi-label classification (MC)* problem $\mathcal{P} = (\mathcal{C}, \mathcal{X})$ consists of a finite set $\mathcal{C}$ of *labels*, denoted by $A_1, A_2, \ldots$, and a finite set $\mathcal{X}$ of pairs (x, y), where $x \in \mathbb{R}^D$ $(D \geq 1)$ is a *data point*, and $y \subseteq \mathcal{C}$ is the *ground truth* of x. The ground truth y *associated with a data point* x characterizes both the *positive* and the *negative labels* associated with x, defined to be y and $\{\neg A : A \in \mathcal{C} \setminus y\}$, respectively. In ROAD-R, a data point corresponds to a bounding box, and each box is labeled with the positive labels representing (i) the agent performing the actions in the box, (ii) the actions being performed, and (iii) the locations where the actions take place. See Appendix A for a detailed description of each label. Consider an MC problem $\mathcal{P} = (\mathcal{C}, \mathcal{X})$. A *prediction* p is a set of positive and negative labels such that for each label $A \in \mathcal{C}$, either $A \in p$ or $\neg A \in p$. A *model* m for $\mathcal{P}$ is a function $m(\cdot, \cdot)$ mapping every label A and every data point x to $[0, 1]$. A data point x is *predicted* by a model m to have label A if its *output value* $m(A, x)$ is greater than a user-defined *threshold* $\theta \in [0, 1]$. The *prediction of model* m *for data point* x is the set $\{A : A \in \mathcal{C}, m(A, x) > \theta\} \cup \{\neg A : A \in \mathcal{C}, m(A, x) \leq \theta\}$ of positive and negative labels.

An *MC problem with propositional logic requirements* $(\mathcal{P}, \Pi)$ consists of an MC problem $\mathcal{P}$ and a finite set Π of propositional logic constraints on the labels of $\mathcal{P}$. Consider an MC problem with propositional logic requirements $(\mathcal{P}, \Pi)$. Each constraint in Π delimits the set of predictions that can be associated with each data point by ruling out those that violate it. A prediction p is *admissible* if each constraint r in Π is *satisfied* by p. A model m for $\mathcal{P}$ *satisfies* (resp., *violates*) the constraints on a data point x if the prediction of m for x is (resp., is not) admissible.

Example 1 The requirement that a traffic light cannot be both red and green corresponds to the constraint $\neg$RedTL $\vee$ $\neg$GreenTL. Any prediction with $\{$RedTL, GreenTL$\}$ is non-admissible. An example of such predictions made by the SOTA models is shown in Fig. 1.

Given an MC problem with propositional logic requirements, it is possible to take advantage of the constraints in two different ways: (i) they can be exploited during learning to teach the model the background knowledge that they express, and (ii) they can be used as post-processing to turn a non-admissible prediction into an admissible one. Models in the first and second category have a *constrained loss (CL)* and *constrained output (CO)*, respectively. Constrained loss models have the advantage that the constraints are deployed during the training phase, and this should result in models (i) with a higher understanding of the problem and a better performance, but still (ii) with no guarantee that no violations will be committed. On the other hand, constrained output models (i) do not exploit the additional knowledge during training, but (ii) are guaranteed to have no violations in the final outputs. These two options are not mutually exclusive (i.e., can be used together), and which one is to be deployed depends also on the extent to which a system is available. For instance, there can be companies that already have their own models (which can be black boxes) and want to make them compliant with a set of requirements without modifying the model itself. On the other hand, the exploitation of the constraints in the learning phase can be an attractive option for those who have a good knowledge of the model and want to further improve it.

3 ROAD-R

ROAD-R extends the ROAD dataset[2] (Singh et al. 2022) by introducing a set Π of 243 constraints that specify the space of admissible outputs.

In order to improve the usability of our dataset, we write each constraint as a disjunction of positive and negative labels, i.e., as expressions having the form:

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

where $n \geq 1$, and each l_i is either a negative label $\neg A$ or a positive label A. Thus, Π can be equivalently seen as a formula in conjunctive normal form (CNF), which is the standard form used by propositional logic solvers. Notice that for any propositional formula there is an equivalent one in CNF.

The requirements have been manually specified following three steps:

1. An initial set of constraints Π_1 was manually created,
2. A subset $\Pi_2 \subset \Pi_1$ was retained by eliminating all those constraints that were entailed by the others,
3. The final subset $\Pi \subset \Pi_2$ was retained by keeping only those requirements that were always satisfied by the ground-truth labels of the entire ROAD-R dataset.

Considering the above procedure, a few considerations are in order:

1. The requirement specification process (i) is a standard step in the development of any software, necessary to characterize the expected behavior of the system and then verify that the system functions as expected; and (ii) deeply involves the stakeholders/designers of the system (see, e.g., (Sommerville 2011)). As a consequence, the set Π_1 is not guaranteed to be complete from every possible point of view. Indeed, with a different set of labels and/or in different contexts, other constraints may hold. For instance, some

[2] Dataset and code are available at: https://github.com/EGiunchiglia/ROAD-R.

Table 2 Constraint statistics

Statistics	
$\mid \mathcal{C} \mid$	41
$\mid \Pi \mid$	243
$\text{avg}_{r \in \Pi}(\mid r \mid)$	2.86
$\mid \{A \in \mathcal{C} : \exists r \in \Pi. A \in r\} \mid$	41
$\mid \{A \in \mathcal{C} : \exists r \in \Pi. \neg A \in r\} \mid$	38
$\min_{A \in \mathcal{C}}(\mid \{r \in \Pi : \{A, \neg A\} \cap r \neq \emptyset\} \mid)$	2
$\text{avg}_{A \in \mathcal{C}}(\mid \{r \in \Pi : \{A, \neg A\} \cap r \neq \emptyset\} \mid)$	16.95
$\max_{A \in \mathcal{C}}(\mid \{r \in \Pi : \{A, \neg A\} \cap r \neq \emptyset\} \mid)$	31

Table 3 Constraint statistics. Π_n is the set of constraints r in Π with $\mid r \mid = n$, i.e., with n positive and negative labels. $\overline{\mathcal{C}} = \{\neg A : A \in \mathcal{C}\}$

n	$\mid \Pi_n \mid$	$\text{avg}_{r \in \Pi_n}(\mid r \cap \overline{\mathcal{C}} \mid)$	$\text{avg}_{r \in \Pi_n}(\mid r \cap \mathcal{C} \mid)$
2	215	1.995	0.005
3	5	1	2
7	1	1	6
8	6	1	7
9	6	1	8
10	1	0	10
12	1	1	11
14	1	0	14
15	7	1	14
Total	243	1.87	0.96

roads can be closed to “large vehicles" and in some countries it is possible to have traffic lights with both the green and amber lights on;

2. The elimination of the constraints that are violated by the ground-truth labels of the entire dataset—despite their validity—is a necessary step in order to maintain (i) consistency between the knowledge provided by the constraints and by the data points, and (ii) backward compatibility with the ROAD dataset. Indeed, some constraints in Π_1, like “it is not possible for an agent to both move towards and move away", have been discarded, since they were not satisfied by all the data points because of errors in the ground-truth labels;
3. Following the standard practice adopted in software development, the requirement specification process should come before the software development begins and before the annotation of the dataset. Indeed, this would have allowed to (i) simplify the annotation process, and then (ii) validate the annotated dataset.

Given the above, ROAD-R, along with the presented models, is a first step pushing in the direction of having a new generation of machine learning models (i) whose design starts with the specification of the requirements that it should satisfy, and (ii) able to learn from and then obey to the constraints. This will help in the deployment of machine learning models in all application domains, including safety-critical ones. Indeed, as stated in

Amodei et al. (2016) and Hoernle et al. (2022), to be applied in such settings models need to be guaranteed to be able to operate within boundaries specified by domain experts.

Tables 2 and 3 give a high-level description of the properties of the set Π of constraints. Notice that, with a slight abuse of notation, in the tables, we use a set-based notation for the requirements. Each requirement of the form (1) thus becomes $\{l_1, l_2, \dots, l_n\}$. Such notation allows us to express the properties of the requirements in a more succinct way. From Table 2, we can see that:

- Aall the constraints have between 2 and 15 positive and negative labels, with an average of 2.86,
- All the labels appear positively in Π.
- Of the 41 labels, 38 appear negatively in Π, and
- Each label appears either positively or negatively between 2 and 31 times in Π, with an average of 16.95.

Table 3 gives a close-up view of structure of the constraints, showing the number of rules having n positive and negative labels, together with the average number of negative and positive labels in such rules. As witnessed by Table 3, in the 243 constraints, there are two in which all the labels are positive (expressing that there must be at least one agent and that every agent but traffic lights has at least one location), and 214 in which all the labels are negative (expressing mutual exclusion between two labels). All the constraints with more than two labels have at most one negative label, as they express a one-to-many relation between actions and agents (like "if something is crossing, then it is a pedestrian or a cyclist"). Constraints like "pedestrians should cross at crossings", which might not be satisfied in practice, are not included. The list with all the 243 requirements, with their natural language explanations, is in Appendix D, Tables 9, 10, and 11. Overall, the 243 requirements restrict the number of admissible prediction to $4985868 \sim 5 \times 10^6$, thus ruling out $(2^{41} - 4985868) \sim 10^{12}$ non-admissible predictions.[3]

4 ROAD-R and SOTA models

As a first step, we ran 6 SOTA temporal feature learning architectures as part of a 3D-RetinaNet model (Singh et al. 2022) (with a 2D-ConvNet backbone made of Resnet50 (He et al. 2016)) for event detection and evaluated to which extent constraints are violated. Each SOTA model takes as input a sequence of frames, and it returns: (i) a set of bounding boxes for each frame, and (ii) a vector $v \in [0, 1]^{|C|}$ for each bounding box. For each bounding box, the final prediction is then the set of positive and negative labels obtained by thresholding v as described in Sect. 2. We considered:

1. *2D-ConvNet (C2D)* (Wang et al. 2018): 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.

[3] The number of admissible predictions has been computed with relsat: https://github.com/roberto-bayardo/relsat/.

2. *Inflated 3D-ConvNet (I3D)* (Carreira and Zisserman 2017): a sequential learning architecture extendable to any SOTA image classification model (2D-ConvNet based), able to learn continuous spatio-temporal features from the sequence of frames.
3. *Recurrent Convolutional Network (RCN)* (Singh and Cuzzolin 2019): 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)* (Hua et al. 2018): 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)* (Hua et al. 2018): 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* (Feichtenhofer et al. 2019): a 3D-CNN architecture that contains both slow and fast pathways for extracting the sequential features. A slow pathway computes the spatial semantics at a low frame rate, while a fast pathway processes high frame rate for capturing the motion features. Both the pathways are fused in a single architecture by lateral connections.

We trained 3D-RetinaNet[4] using the same hyperparameter settings for all the models: (i) batch size equal to 4, (ii) sequence length equal to 8, and (iii) image input size equal to 512×682. All the models were initialized with the Kinetics pre-trained weights. An SGD optimizer (LeCun et al. 2012) with step learning rate was used. The initial learning rate was set to 0.0041 for all the models except SlowFast, for which it was set to 0.0021 due to the diverse nature of slow and fast pathways. All the models were trained for 30 epochs, and the learning rate was made to drop by a factor of 10 after 18 and 25 epochs. The machine used for the experiments has 64 CPUs (2.2 GHz each) and 4 Titan RTX GPUs having 24 GB of RAM each.

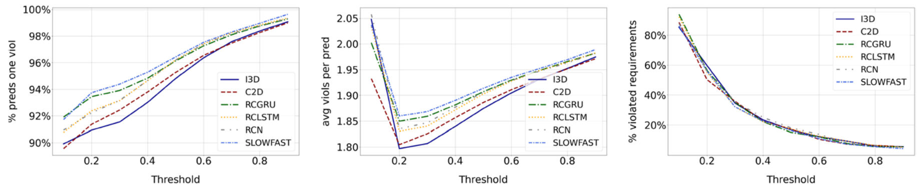
To measure the models' performance, we used the *frame mean average precision* (f-mAP), which is the standard metric used for action detection (see, e.g., (Kalogeiton et al. 2017; Li et al. 2018)) and is obtained by calculating for each class the mean average precision over all frames, averaging the final results as shown in Eq. (2). In our experiments, we set IoU threshold equal to 0.5 and 0.75, indicated as f-mAP@0.5 and f-mAP@0.75, respectively.

$$\text{f-mAP@}\tau = \frac{1}{|\mathcal{C}|}\frac{1}{F}\sum_{i=1}^{|\mathcal{C}|}\sum_{j=1}^{F}\text{AP}_{ij}, \tag{2}$$

where F is the number of frames, and AP_{ij} is the average precision for class i at frame j at IoU τ. The results for the SOTA models at IoU threshold 0.5 and 0.75 are reported in Table 4, column "SOTA".

To measure the extent to which each system violates the constraints, we used the following metrics:

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



(a) Percentage of predictions violating at least one constraint.

(b) Average number of violations committed per prediction.

(c) Percentage of constraints violated at least once.

Fig. 2 ROAD-R and SOTA models. In the x-axis, there is the threshold $\theta \in [0.1, 0.9]$, step 0.1

- The percentage of non-admissible predictions,
- The average number of violations committed per prediction, and
- The percentage of constraints violated at least once,

while varying the threshold θ from 0.1 to 0.9 with step 0.1. The results are in Fig. 2, where (to improve readability) we do not plot the values corresponding to $\theta = 0.0$ and $\theta = 1.0$. For $\theta = 0.0$ (resp., $\theta = 1.0$), all the predictions are positive (resp., negative), and thus the corresponding values are (in order) 100%, 214, and 214/243 (resp., 100%, 2, and 2/243).

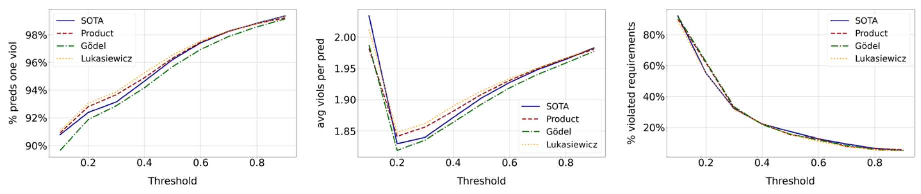
Consider the results in Table 4, column "SOTA", and in Fig. 2. First, note that the performances are not an indicator of the ability of the model to satisfy the constraints. Indeed, higher f-mAPs do not correspond to lower trends in the plots of Fig. 2b. For example, RCGRU performs better than C2D for both IoU = 0.5 and IoU = 0.75, however, its curve is above C2D's in both Fig. 2a and b. Then, note that the percentage of non-admissible predictions is always very high for every model: at its minimum, for $\theta = 0.1$, more than 90% of the predictions are non-admissible, and this percentage reaches 99% for $\theta = 0.9$ (see Fig. 2a). In addition, most predictions violate roughly two constraints, as shown by Fig. 2b. Considering that we are in an autonomous vehicle setting, such results are critical: one of the constraints that is violated by all the baseline models is $\{\neg\text{RedTL}, \neg\text{GreenTL}\}$, corresponding to predictions stating that there is a traffic light with both the red and the green lights on. Figure 1 shows an image where such a prediction is made by C2D. Appendix C contains images with all the models making predictions violating $\{\neg\text{RedTL}, \neg\text{GreenTL}\}$ and other constraints.

5 ROAD-R and CL, CO, and CLCO models

We now show how it is possible to build CL, CO, and CLCO models. In particular, we show how to equip the 6 considered SOTA models with a constrained loss and/or a constrained output. As anticipated in the introduction, we introduce (i) three different methods to build the constrained loss, (ii) three different methods to obtain the constrained output, and (iii) three combinations of constrained loss and constrained output. Thus, we get 9 models for each SOTA model, for a total of 54. In order to get an overall view of the performance gains produced by each method, we also report the average ranking of the 9 proposed methods and SOTA (Demsar 2006), computed as follows: (i) for each row in Table 4, we rank the performances of the 9 CL, CO,

Table 4 f-mAP@0.5 (top table) and f-mAP@0.75 (bottom table) of the (i) current SOTA models, (ii) CL model, (iii) CO models, and (iv) CLCO models. P, G, and L stand for the Product, Gödel, and Łukasiewicz t-norm, respectively. MD, AP, and AP $\times$ O indicate the Minimal Distance policy, Average Precision-based policy and Average Precision and Output-based policy, respectively. In parenthesis we report the difference in performance between of each model and the relative SOTA model. The values of the threshold θ (for the CO and CLCO models) and of the hyperparameter α (for the CL and CLCO models) are given in Table 8 in Appendix B. Best results are in bold

Model	SOTA	Constrained loss (CL)			Constrained output (CO)			Constrained loss and output (CLCO)		
		Product	Gödel	Łukasiewicz	MD	AP	AP×O	(P, AP×O)	(G, AP×O)	(L, AP×O)
C2D	27.57	27.90 (+0.33)	27.97 (+0.40)	27.57 (0.00)	27.41 (−0.16)	27.67 (+0.10)	27.69 (+0.12)	27.90 (+0.33)	**28.16 (+0.59)**	27.86 (+0.29)
I3D	30.12	30.79 (+0.67)	29.98 (−0.14)	31.03 (+0.91)	29.86 (−0.26)	30.15 (+0.03)	30.15 (+0.03)	30.77 (+0.65)	30.02 (−0.10)	**31.21 (+1.09)**
RCGRU	30.78	30.67 (−0.11)	29.57 (−1.21)	31.78 (+1.00)	30.56 (−0.22)	30.78 (0.00)	30.78 (0.00)	30.79 (+0.01)	29.83 (−0.95)	**31.81 (+1.03)**
RCLSTM	30.49	30.48 (−0.01)	30.88 (+0.39)	31.63 (+1.14)	30.09 (−0.40)	30.51 (+0.02)	30.51 (+0.02)	30.50 (+0.01)	30.90 (+0.41)	**31.65 (+1.16)**
RCN	29.64	30.31 (+0.67)	30.24 (+0.60)	31.02 (+1.38)	29.39 (−0.25)	29.80 (+0.16)	29.82 (+0.18)	30.34 (+0.70)	30.28 (+0.64)	**31.02 (+1.38)**
SlowFast	28.79	28.54 (−0.25)	28.91 (+0.12)	28.47 (−0.32)	28.51 (−0.28)	28.86 (+0.07)	28.86 (+0.07)	28.67 (−0.12)	**28.98 (+0.19)**	28.56 (−0.23)
Avg. Rank	7.08	5.75	5.50	4.25	9.50	5.75	5.42	4.42	4.50	**2.83**
C2D	12.66	13.04 (+0.38)	12.90 (+0.24)	12.30 (−0.36)	12.62 (−0.04)	12.72 (+0.06)	12.72 (+0.06)	**13.05 (+0.39)**	13.01 (+0.35)	12.55 (−0.11)
I3D	16.29	16.47 (+0.18)	16.19 (−0.10)	16.32 (+0.03)	16.22 (−0.07)	16.31 (+0.02)	16.31 (+0.02)	**16.48 (+0.19)**	16.20 (−0.09)	16.42 (+0.13)
RCGRU	15.98	16.57 (+0.59)	15.39 (−0.59)	17.26 (+1.28)	15.97 (−0.01)	16.02 (+0.04)	16.03 (+0.05)	16.63 (+0.65)	15.63 (−0.35)	**17.27 (+1.29)**
RCLSTM	16.64	16.14 (−0.50)	15.65 (−0.99)	16.34 (−0.30)	16.38 (−0.26)	**16.69 (+0.05)**	16.66 (+0.02)	16.15 (−0.49)	15.67 (−0.97)	16.39 (−0.25)
RCN	15.82	16.57 (+0.75)	16.17 (+0.35)	17.25 (+1.43)	15.73 (−0.09)	15.87 (+0.05)	15.89 (+0.07)	16.56 (+0.74)	16.19 (+0.37)	**17.26 (+1.44)**
SlowFast	14.40	14.84 (+0.44)	15.38 (+0.98)	13.80 (−0.60)	14.33 (−0.07)	14.47 (+0.07)	14.47 (+0.07)	14.94 (+0.54)	**15.42 (+1.02)**	13.84 (−0.56)
Avg. Rank	6.67	3.83	7.00	5.67	7.83	5.00	4.83	**3.17**	6.00	4.50



(a) Percentage of predictions violating at least one constraint.

(b) Average number of violations committed per prediction.

(c) Percentage of constraints violated at least once.

Fig. 3 Comparison of the behaviour of RCLSTM and CL-RCLSTM (with Product, Gödel and Łukasiewicz loss) with respect to the requirements. In the x-axis, there is the threshold $\theta \in [0.1, 0.9]$, step 0.1

and CLCO models and of the SOTA model separately: the best performing model gets the rank 1, the second best gets rank 2, etc., and in case of ties, the rank is split (e.g., the assigned rank is 1.5 if two models have the best performance), and (ii) for each column, we take the average of the rankings computed in step 1. See Table 4 for f-mAP@0.5, f-mAP@0.75 and average rankings, where, for each row the best results are in bold. The details of the implemented models with constrained loss, constrained output, and both constrained loss and constrained output is given in the three subsections below.

5.1 Constrained loss

To constrain the loss, we take inspiration from the approaches proposed in Diligenti et al. (2017a, 2017b), and we train the models using the standard localization and classification losses, to which we add a regularization term. This last term represents the degree of satisfaction of the constraints in Π and has the form:

$$\mathcal{L}_\Pi = \alpha \sum_{i=1}^{|\Pi|}(1 - t(r_i)),$$

where r_i represents the ith constraint in Π, $t(r_i)$ represents the fuzzy logic relaxation of r_i, and α is a hyperparameter ruling the weight of the regularization term (the higher α is, the more relevant the term corresponding to the constraints becomes, up to the limit case in which $\alpha \to \infty$, and the constraints become hard (Diligenti et al. 2017b)). We considered $\alpha \in \{1, 10, 100\}$ and the three fundamental t-norms: (i) Product t-norm, (ii) Gödel t-norm, and (iii) Łukasiewicz t-norm as fuzzy logic relaxations (Hájek 1998). The best results for f-mAP@0.5 and f-mAP@0.75 while varying α are in Table 4, columns Product, Gödel, and Łukasiewicz. As can be seen, SOTA never achieves the best average ranking, even when compared with only the three CL methods. Of these, Łukasiewicz (for IoU = 0.5) and Product (for IoU = 0.75) have the best ranking, though for some model and IoU, the best performances are obtained with Gödel. In only one case (for RCLSTM at IoU = 0.75), the SOTA model performs better than the CL models. Furthermore, we measure the extent to which the CL models violate the constraints using the metrics introduced in the previous section, and we never get any significant reduction in the number of predictions violating the constraints. As example, we plot the resulting charts in Fig. 3 for the SOTA model RCLSTM and the CL-RCLSTM models with Product, Gödel and Łukasiewicz loss. As it can be seen from Fig. 3a, the CL models' predictions also violate the constraints at least 90% of the times.

5.2 Constrained output

We now consider the problem of how to correct a prediction p whose admissibility is evaluated at a given threshold θ. The first observation is that determining the existence of an admissible prediction is an intractable problem: indeed, this is just a reformulation of the satisfiability problem in propositional logic, which is well known to be NP-complete. Despite this, we want to correct any non-admissable prediction p in such a way that (i) the final prediction is admissible, and (ii) the performance of the final model either improves or remains unaltered.

In order to achieve the above, we first test the policy of trying to correct as few labels as possible. More precisely, for each prediction q, $(p \setminus q)$ is the set of positive and negative predictions on which q differs from p. Then, we can compute the admissible prediction q with the minimum number of differences, i.e., such that $| p \setminus q |$ is minimal. We call such policy *Minimal Distance (MD)*. Unfortunately, no polynomial time algorithm is known to solve this problem.

Theorem 1 *Let* $(\mathcal{P}, \Pi)$ *be an MC problem with requirements. Let p be a prediction. For each positive d, determining the existence of an admissible prediction q such that* $| p \setminus q | \leq d$ *is an NP-complete problem.*

The theorem is an easy consequence of Proposition 1 in Bailleux and Marquis (2006). In order to be able to solve the problem in practice, we formulate the problem of finding an admissible prediction with minimal $| p \setminus q |$ as a weighted partial maximum satisfiability (PMaxSAT) problem of a set of clauses (see, e.g., (Li and Manyà 2009)) in which

1. Each constraint in Π corresponds to a clause marked as hard, and
2. Each positive and negative prediction in p corresponds to a unit clause marked as soft with unitary weight.

In our setting, a *clause* is a disjunction of literals, and a *literal* is either a positive label or its negation, representing the corresponding negative label. A clause is *unit* if it consists of a single literal. This allows us to use the very efficient solvers publicly available for PMaxSat problems. In particular, in our experiments, we used MaxHS (Hickey and Bacchus 2019), and running times were in the order of 10^{-3}s at most. As intended, since we assign all labels unitary weight, MaxHS returns the admissible prediction q with as few as possible labels flipped. Notice that flipping the ith label amounts to changing its output value o_i from a value below the threshold to another value $f(o_i)$ above the threshold or vice versa. In all our experiments, we considered (i) $f(o_i) = \theta + \epsilon$, if $o_i < \theta$, and (ii) $f(o_i) = \theta - \epsilon$, otherwise ($\epsilon = 10^{-3}$). In this way, assuming $o_i \leq \theta$ (i.e., the label is negatively predicted and then flipped positive), we expect $f(o_i)$ to be lower (but still higher than θ) than the output values of the non-flipped, positively predicted labels. It is done analogously for the case $o_i > \theta$. We tested this approach with all the thresholds θ from 0.1 to 0.9 with step 0.1, and the resulting f-mAP@0.5 and f-mAP@0.75 for the best threshold are reported in Table 4, column MD. As we can see from the table, despite the fact that we are minimizing the number of corrections, the results obtained by the CO-MD model are always worse than the ones obtained by the SOTA models. We can hence conclude that adding such post-processing has a detrimental effect on the models' performance.

Thus, we need alternative policies to correct a non-admissible prediction p. We generalize the problem by assigning a positive weight w_i (representing the cost of correcting the ith label in p) and then computing the admissible prediction q that minimizes $\sum_{i=1}^{|\mathcal{C}|} w_i$. More precisely, for every prediction q, $cost(p, q) = \sum_{i \in \mathcal{I}} w_i$, $\mathcal{I}$ being the set of indexes of the labels in $(p \setminus q)$. Then, we can compute the admissible prediction q such that $cost(p, q)$ is minimal. As it is a generalization of the problem above, no polynomial-time algorithm is known to solve this problem.

Theorem 2 *Let $(\mathcal{P}, \Pi)$ be an MC problem with requirements. Let p be a prediction, and let w_i be the cost of correcting the ith label in p. For each positive d, determining the existence of an admissible prediction q such that $cost(p, q) \leq d$ is an NP-complete problem.*

This is an easy consequence of Theorem 1. We can again formulate the problem as a PMaxSAT problem in which:

1. Each constraint in Π corresponds to a clause marked as hard, and
2. For each i $(1 \leq i \leq | \mathcal{C} |)$, the prediction in p for the ith label corresponds to a unit clause having weight w_i.

Given the above formulation, we tested two different policies for choosing the weight associated to each label:

1. *Average Precision-based (AP)*, in which each w_i is equal to the average precision AP_i of the ith label, and
2. *Average Precision and Output-based* (AP×O), in which each $w_i = \mathrm{AP}_i \times c_i$, where c_i is equal to (i) the output o_i of the model for the ith label if $o_i > \theta$, and (ii) $(1 - o_i)$, otherwise.

Differently from the MD policy, in order to take a decision, these policies take into account the reliability of the output o_i for the ith label. We again tested the two policies with all the thresholds θ from 0.1 to 0.9 with step 0.1, and the resulting f-mAP@0.5 and f-mAP@0.75 for the best threshold are reported in Table 4, columns AP, and AP×O. Comparing the predictions of the SOTA models with the CO-AP and the CO-AP×O models, we can see that flipping the variables taking into account the average precision (i) never leads to worse performances than the ones of the SOTA models, and (ii) for IoU = 0.75, correcting the output of RCLSTM with AP and AP×O gives the best and second best performance in the row. Notice that the differences in the performances between AP and AP×O are negligible, AP×O being better than AP more often. The average rankings are in line with the above statements.

5.3 Constrained loss and output

Given the results presented in the previous paragraph, of the 9 possible combinations of a constrained loss and a constrained output, we consider only the ones with AP×O as constrained output. The results are shown in Table 4, last three columns. Given the constant improvement produced by the constrained output AP×O over the SOTA models discussed in the previous paragraph, the results of the CLCO models are not surprising: post-processing the output with AP×O policy produces again relatively small but almost always constant improvements over the corresponding CL models. The average rankings are in line with the above statements.

Considering the results in Table 4 all together, we see that

1. Constraining the output alone guarantees the compliance with the constraints, but improvements in the performances are constant but limited,
2. Constraining the loss alone does not guarantee the satisfaction of the requirements but can lead to non marginal improvements in the performances,
3. The best performances (the numbers in bold) are always obtained by constraining the output, and thus it is always possible to (i) improve the performance of each SOTA model, and (ii) guarantee to be compliant with the requirements,
4. On average, the best performances are obtained by CLCO models, as witnessed by the average rankings,
5. The best performing model is CLCO-RCGRU, i.e., RCGRU with Łukasiewicz constrained loss and AP×O constrained output: such model (i) is compliant with the constraints by construction, and (ii) has f-mAP = 31.81 for IoU = 0.5, and f-mAP = 17.27 for IoU = 0.75. RCGRU (without CL and CO) (i) produces predictions that violate the constraints at least 92% of the times, and (ii) has f-mAP = 30.78 for IoU = 0.5, and f-mAP = 15.98 for IoU = 0.75.

6 Related work

The approach proposed in this paper generalizes HMC problems, in which requirements are binary and have the form $(A \to B)$, corresponding to our $(\neg A \vee B)$. Many models have been developed for HMC; see, e.g., (Vens et al. 2008; Wehrmann et al. 2018; Giunchiglia and Lukasiewicz 2020).

Interestingly, when dealing with more complex logical requirements on the output space, in the past, researchers have mostly focused on exploiting the background knowledge that they express to improve performance and/or to deal with data scarcity, curiously neglecting the problem of guaranteeing their satisfaction. Many works go in this direction, such as Hu et al. (2016a, 2016b), where an iterative method to embed structured logical information into the neural networks' weights is introduced: at each step, the authors consider a *teacher network* based on the set of logical rules to train a *student network* to fit both supervisions and logic rules. Another neural model is considered in Li and Srikumar (2019), in which some neurons are associated with logical predicates, and their activation is modified on the ground of the activation of the neurons corresponding to predicates that co-occur in the same rules. An entire line of research is dedicated to embedding logical constraints into loss functions; see, e.g., (Diligenti et al. 2017b; Donadello et al. 2017; Xu et al. 2018). These works consider a fuzzy relaxation of FOL formulas to get a differentiable loss function that can be minimized by gradient descent. However, in all the above methods, there is no guarantee that the constraints will be actually satisfied. Recently, this problem has gained more relevance, and few works now propose novel ways of addressing the problem. One of such works is (Giunchiglia and Lukasiewicz 2021), which presents a novel neural model called *coherent-by-construction network* (CCN). CCN not only exploits the knowledge expressed by the constraints, but is also able to guarantee the constraints' satisfaction. However, that model is able to guarantee the satisfaction of constraints written as normal logic rules with at least one positive label, and thus is not able to deal with all the ROAD-R's requirements. Another work that goes in this direction is (Dragone et al.

2021), in which NESTER is proposed. In this case, the constraints are not mapped into the last layer of the network (like CCN), but they are enforced by passing the outputs of the neural network to a constraint program, which enforces the constraints. The most recent work is given by Hoernle et al. (2022), where the authors propose MultiPlexNet. MultiplexNet can impose constraints consisting of any quantifier-free linear arithmetic formula over the rationals (thus, involving “+", “≥", “¬", “∧", and “∨"). In order to train the model with such constraints, the formulas are firstly expressed in disjunctive normal form (DNF), and then the output layer of the network is augmented to include a separate transformation for each term in the DNF formula. Thus, the network’s output layer can be viewed as a multiplexor in a logical circuit that permits for a branching of logic. For a general overview of deep learning with logical constraints, see the survey (Giunchiglia et al. 2022).

In the video understanding field, some recent works have started to argue the importance of being able to extract structured information from videos and to incorporate background knowledge in the models. For example, (Curtis et al. 2020) propose a challenge to test the models’ ability of extracting knowledge graphs from videos. In Mahon et al. (2020), the authors develop a model that is able to exploit the knowledge expressed in logical rules to extract knowledge graphs from videos. However, ROAD-R is the first dataset which proposes the incorporation of logical constraints into deep learning models for videos, and thus represents a truly novel challenge.

7 Summary and outlook

In this paper, we proposed a new learning framework, called learning with requirements, and a new dataset for this task, called ROAD-R. We showed that SOTA models most of the times violate the requirements, and how it is possible to exploit the requirements to create models that are compliant with (i.e., strictly satisfy) the requirements while improving their performance.

ROAD-R opens up a number of research possibilities. The most straightforward open problem is how to create neural-based models that are compliant by design with the given requirements, i.e., without the need of any post-processing steps. However, other directions are also possible. For example, it is an open question whether the annotated constraints can help in alleviating the data greediness characteristic of the large deep learning models usually deployed in the autonomous driving setting. Indeed, we can now use the requirements to train models on both labelled and unlabelled data. Another open question is whether neural models, in addition to bounding boxes and labels, can also learn the requirements that we annotated. In this case, the annotated requirements could be used to measure the coverage of the learned ones.

Finally, in the future, we will further extend ROAD-R. In particular, we plan to annotate ROAD with temporal constraints stating facts like “a traffic light becomes red after being green”, and with soft constraints stating likely facts like “pedestrians should cross at crossings”.

Appendix A: ROAD labels

Here, we provide a detailed description of the meaning of each of the 41 labels. In particular, we describe (i) the labels associated with agents in Table 5, (ii) the labels associated with actions in Table 6, and (iii) the labels associated with locations in Table 7. In the last column of each table, we also report the abbreviations used in our implementation and

Table 5 Agent labels with descriptions and abbreviations Singh et al. (2022)

Label name	Description	Abbrv.
Pedestrian	A person including children	Ped
Car	A car up to the size of a multi-purpose vehicle	Car
Cyclist	A person is riding a push/electric bicycle	Cyc
Motorbike	Motorbike, dirt bike, scooter with 2/3 wheels	Mobike
Medium vehicle	Vehicle larger than a car, such as van	MedVeh
Large vehicle	Vehicle larger than a van, such as a lorry	LarVeh
Bus	A single or double-decker bus or coach	Bus
Emergency vehicle	Ambulance, police car, fire engine, etc	EmVeh
AV traffic light	Traffic light related to the AV lane	TL
Other traffic light	Traffic light not related to the AV lane	OthTL

associated with each label. Such abbreviations were taken from https://github.com/gurkirt/3D-RetinaNet, and allowed a seamless integration of our code with the SOTA models' code. The abbreviations are used in Tables 9, 10, and 11 to write the requirements using the set-based notation.

Table 6 Action labels with descriptions and abbreviations Singh et al. (2022)

Label name	Description	Abbrv.
Move away	Agent moving in a direction that increases the distance between Agent and AV	MovAway
Move towards	Agent moving in a direction that decreases the distance between Agent and AV	MovTow
Move	Agent moving perpendicular to the traffic flow or vehicle lane	Mov
Brake	Agent is slowing down, vehicle braking lights are lit	Brake
Stop	Agent stationary but in ready position to move	Stop
Indicating left	Agent indicating left by flashing left indicator light, or using a hand signal	IncatLeft
Indicating right	Agent indicating right by flashing right indicator light, or using a hand signal	IncatRht
Hazard lights on	Hazards lights are flashing on a vehicle	HazLit
Turn left	Agent is turning in left direction	TurLft
Turn right	Agent is turning in right direction	TurRht
Overtake	Agent is moving around a slow-moving user, often switching lanes to overtake	Ovtak
Wait to cross	Agent on a pavement, stationary, facing in the direction of the road	Wait2X
Cross road from left	Agent crossing road, starting from the left and moving towards the right of AV	XingFmLft
Cross road from right	Agent crossing road, starting from the right and moving towards the left of AV	XingFmRht
Crossing	Agent crossing road	Xing
Push object	Agent pushing object, such as trolley or pushchair, wheelchair or bicycle	PushObj
Red traffic light	Traffic light with red light lit	Red
Amber traffic light	Traffic light with amber light lit	Amber
Green traffic light	Traffic light with green light lit	Green

Table 7 Location labels with descriptions and abbreviations Singh et al. (2022)

Labelname	Description	Abbrv.
AV lane	Agent in same road lane as AV	VehLane
Outgoing lane	Agent in road lane that should be flowing in the same direction as vehicle lane	OutgoLane
Outgoing cycle lane	Agent in the cycle lane that should be flowing in the same direction as AV	OutgoCycLane
Incoming lane	Agent in road lane that should be flowing in the opposite direction as vehicle lane	IncomLane
Incoming cycle lane	Agent in the cycle lane that should be flowing in the opposite direction as AV	IncomCycLane
Pavement	A pavement that is perpendicular to the movement of the AV	Pav
Left pavement	Pavement to the left side of AV	LftPav
Right pavement	Pavement to the right side of AV	RhtPav
Junction	Road linked	Jun
Crossing location	A marked section of road for cross, such as zebra or pelican crossing	XingLoc
Bus stop	A marked bus stop area on road, or a section of pavement next to a bus stop sign	BusStop
Parking	A marked parking area on the side of the road	Parking

Appendix B: Experimental setup

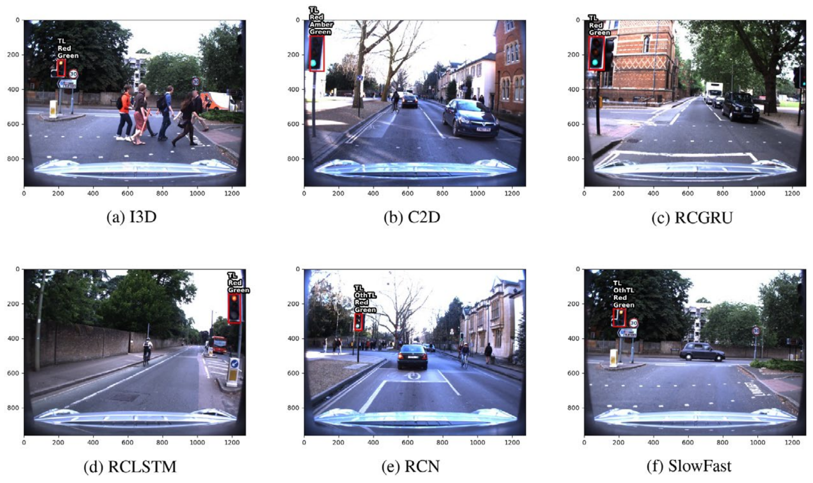
In Table 8, we report the values of θ and α used in our experimental analysis.

Table 8 Values of α, θ and both α and θ for CL, CO, and CLCO models, respectively (values for f-mAP@0.5 above and for f-mAP@0.75 below)

Model	CL			CO			CLCO		
	P	G	L	MD	AP	AP×O	(P, AP×O)	(G, AP×O)	(L, AP×O)
C2D	1	1	10	0.4	0.9	0.9	(1, 0.7)	(1, 0.8)	(10, 0.9)
I3D	10	10	10	0.4	0.5	0.5	(10, 0.9)	(10, 0.7)	(10, 0.6)
RCGRU	10	1	10	0.4	0.5	0.7	(10, 0.6)	(1, 0.9)	(10, 0.5)
RCLSTM	10	1	10	0.4	0.7	0.7	(10, 0.5)	(1, 0.8)	(10, 0.9)
RCN	10	10	10	0.4	0.5	0.5	(10, 0.6)	(10, 0.8)	(10, 0.5)
SlowFast	10	1	10	0.5	0.9	0.9	(10, 0.9)	(1, 0.5)	(10, 0.7)
C2D	1	1	10	0.4	0.9	0.9	(1, 0.8)	(1, 0.8)	(10, 0.9)
I3D	10	10	10	0.5	0.8	0.8	(10, 0.6)	(10, 0.6)	(10, 0.6)
RCGRU	10	1	10	0.4	0.5	0.5	(10, 0.6)	(10, 0.9)	(10, 0.6)
RCLSTM	10	1	10	0.4	0.5	0.7	(10, 0.7)	(10,0.8)	(10, 0.8)
RCN	10	10	10	0.4	0.8	0.5	(10, 0.5)	(10, 0.9)	(10, 0.9)
SlowFast	10	1	10	0.5	0.9	0.9	(10, 0.9)	(10, 0.5)	(10, 0.7)

Appendix C: Visual examples of violations

In this section, we give some visual examples of violations. In order to show different situations in which a non-admissible prediction occurs, we consider various requirements, and (for each of them) we display a non-admissible prediction made by each SOTA model. For all examples, we pick the threshold $\theta = 0.5$, as it is the most intuitive



(a) I3D (b) C2D (c) RCGRU (d) RCLSTM (e) RCN (f) SlowFast

Fig. 4 Examples of violations of {¬RedTL, ¬GreenTL}

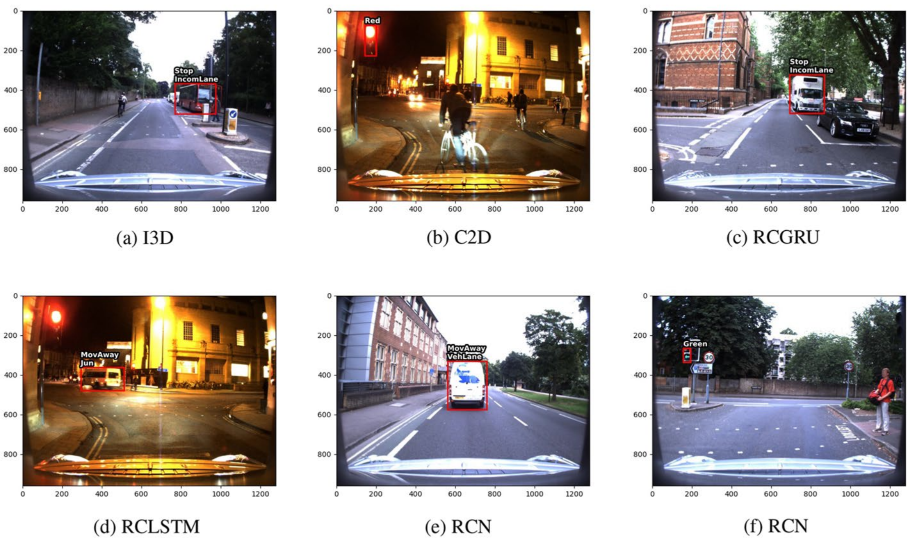


Fig. 5 Examples of violations of {Ped, Car, Cyc, Mobike, MedVeh, LarVeh, Bus, EmVeh, TL, OthTL}

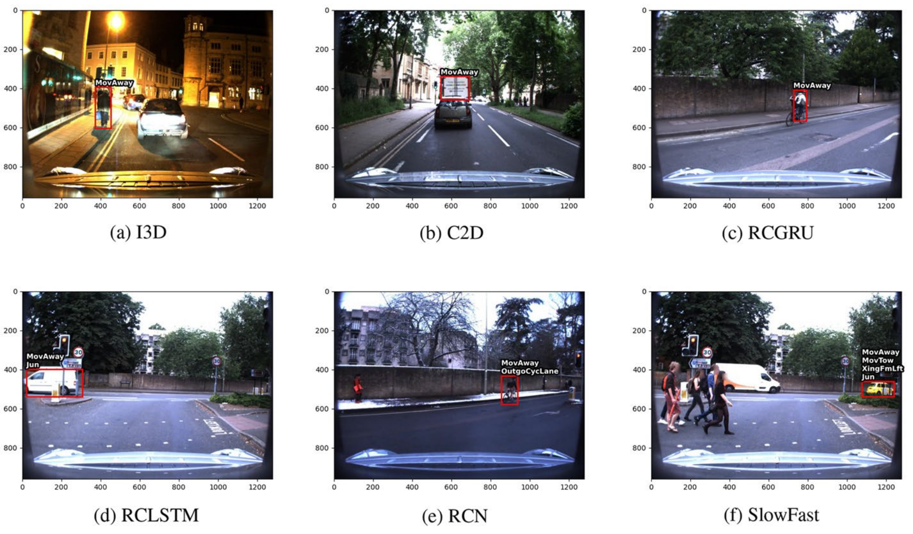


Fig. 6 Examples of violations of {Ped, Car, Cyc, Mobike, MedVeh, LarVeh, Bus, EmVeh, ¬MovAway}

and used threshold in multi-label classification problems. In particular, we show the violations for:

1. {¬RedTL, ¬GreenTL} in Fig. 4,
2. {Ped, Car, Cyc, Mobike, MedVeh, LarVeh, Bus, EmVeh, TL, OthTL} in Fig. 5,
3. {Ped, Car, Cyc, Mobike, MedVeh, LarVeh, Bus, EmVeh, ¬MovAway} in Fig. 6.

In the above list, there is at least one requirement with (i) all negative labels, (ii) all positive labels, and (iii) at least one positive and one negative label.

Appendix D: Requirements

In Tables 9, 10, and 11, we report the list of all the 243 requirements together with their natural language explanations.

Table 9 Requirements table

Requirements	Natural language explanations
{Ped, not PushObj}	If an agent pushes an object then it is a pedestrian
{PushObj, not Ped, MovAway, MovTow, Mov, Stop, TurLft, TurRht, Wait2X, XingFmLft, XingFmRht, Xing}	A pedestrian can only push objects, move away, etc
{Ped, not XingFmLft, Car, Cyc, Mobike, MedVeh, LarVeh, Bus, EmVeh}	Only pedestrains, cars, cyclists, etc. can cross from left
{Ped, not Wait2X, Cyc}	Only pedestrians and cyclists can wait to cross
{Ped, not Stop, Car, Cyc, Mobike, MedVeh, LarVeh, Bus, EmVeh}	Only pedestrians, cars, cyclists, etc can stop
{Ped, not Mov, Car, Cyc, Mobike, MedVeh, LarVeh, Bus, EmVeh}	Only pedestrians, cars, cyclists, etc can move
{Ped, not MovTow, Car, Cyc, Mobike, MedVeh, LarVeh, Bus, EmVeh}	Only pedestrians, cars, cyclists, etc can move towards
{Ped, not MovAway, Car, Cyc, Mobike, MedVeh, LarVeh, Bus, EmVeh}	Only pedestrians, cars, cyclists, etc can move away
{Ovtak, not EmVeh, MovAway, MovTow, Mov, Brake, Stop, IncatLeft, IncatRht, HazLit, TurLft, TurRht, XingFmRht, XingFmLft, Xing}	An emergency vehicle can only overtake, move away etc
{EmVeh, not HazLit, Car, MedVeh, LarVeh, Bus, Mobike}	Only emergency vehicles, cars etc. can have hazards lights on
{Ovtak, not Bus, MovAway, MovTow, Mov, Brake, Stop, IncatLeft, IncatRht, HazLit, TurLft, TurRht, XingFmRht, XingFmLft, Xing}	A bus can only overtake, move away move towards etc
{Ovtak, not MedVeh, MovAway, MovTow, Mov, Brake, Stop, IncatLeft, IncatRht, HazLit, TurLft, TurRht, XingFmRht, XingFmLft, Xing}	A medium vehicle can only overtake, move away, move towards etc
{Ovtak, not LarVeh, MovAway, MovTow, Mov, Brake, Stop, IncatLeft, IncatRht, HazLit, TurLft, TurRht, XingFmRht, XingFmLft, Xing}	A large vehicle can only overtake, move away, move towards etc
{OthTL, not Green, TL}	Only traffic lights and other traffic lights can give a green signal
{OthTL, not Amber, TL}	Only traffic lights and other traffic lights can give an amber signal
{OthTL, not Red, TL}	Only traffic lights and other traffic lights can give a red signal
{Ovtak, not Mobike, MovAway, MovTow, Mov, Brake, Stop, IncatLeft, IncatRht, HazLit, TurLft, TurRht, XingFmRht, XingFmLft, Xing}	A motorbike can only overtake, move away, move towards etc
{Xing, not Cyc, MovAway, MovTow, Mov, Brake, Stop, IncatLeft, IncatRht, TurLft, TurRht, Ovtak, Wait2X, XingFmLft, XingFmRht}	A cyclist can only cross, move away, move towards etc
{Cyc, not Ovtak, MedVeh, LarVeh, Bus, Mobike, EmVeh, Car}	Only cyclists, medium vehicles, large vehicles etc. can overtake
{Cyc, not IncatRht, MedVeh, LarVeh, Bus, Mobike, EmVeh, Car}	Only cyclists, medium vehicles, large vehicles etc. can indicate right
{Cyc, not IncatLeft, MedVeh, LarVeh, Bus, Mobike, EmVeh, Car}	Only cyclists, medium vehicles, large vehicles etc. can indicate left

Table 9 (continued)

Requirements	Natural language explanations
{Cyc, not Brake, MedVeh, LarVeh, Bus, Mobike, EmVeh, Car}	Only cyclists, medium vehicles, large vehicles etc. can brake
{Ovtak, not Car, MovAway, MovTow, Mov, Brake, Stop, IncatLeft, IncatRht, HazLit, TurLft, TurRht, XingFmRht, XingFmLft, Xing}	A car can only overtake, move away, move towards etc
{Car, not TurRht, Cyc, Mobike, MedVeh, LarVeh, Bus, EmVeh}	Only cyclists, medium vehicles, large vehicles etc. can turn right
{Car, not TurLft, Cyc, Mobike, MedVeh, LarVeh, Bus, EmVeh}	Only cyclists, medium vehicles, large vehicles etc. can turn left
{VehLane, OutgoLane, OutgoCycLane, IncomLane, IncomCycLane, Pav, LftPav, RhtPav, Jun, XingLoc, BusStop, Parking, TL, OthTL}	Every agent but traffic lights must have a position
{Ped, Car, Cyc, Mobike, MedVeh, LarVeh, Bus, EmVeh, TL, OthTL}	There must be at least an agent
{not Car, not Cyc}	A car cannot be a cyclist
{not Car, not Mobike}	A car cannot be a motorbike
{not Car, not MedVeh}	A car cannot be a medium vehicle
{not Car, not LarVeh}	A car cannot be a large vehicle
{not Car, not Bus}	A car cannot be a bus
{not Cyc, not Mobike}	A cyclist cannot be a motorbike
{not Car, not EmVeh}	A car cannot be a emergency vehicle
{not Car, not TL}	A car cannot be a traffic light
{not Cyc, not MedVeh}	A cyclist cannot be a medium vehicle
{not Car, not OthTL}	A car cannot be a other traffic light
{not Car, not Red}	A car cannot signal red
{not Cyc, not LarVeh}	A cyclist cannot be a large vehicle
{not Car, not Amber}	A car cannot signal amber
{not Car, not Green}	A car cannot signal green
{not Cyc, not Bus}	A cyclist cannot be a bus
{not Mobike, not MedVeh }	A motorbike cannot be a medium vehicle
{not Cyc, not EmVeh}	A cyclist cannot be an emergency vehicle
{not Mobike, not LarVeh}	A motorbike cannot be a large vehicle

Table 9 (continued)

Requirements	Natural language explanations
{not Cyc, not TL}	A cyclist cannot be a traffic light
{not Cyc, not OthTL}	A cyclist cannot be a other traffic light
{not Mobike, not Bus}	A motorbike cannot be a bus
{not Cyc, not Red}	A cyclist cannot signal red
{not MedVeh, not LarVeh}	A medium vehicle cannot be a large vehicle
{not Mobike, not EmVeh}	A motorbike cannot be an emergency vehicle
{not Cyc, not Amber}	A cyclist cannot signal amber
{not Car, not Wait2X}	A car cannot wait to cross
{not TL, not TurLft}	A traffic light cannot turn left
{not Green, not MovTow}	A green traffic light cannot move towards
{not Red, not Stop}	A red traffic light cannot stop
{not OthTL, not IncatRht}	A other traffic light cannot indicate right
{not TurRht, not TL}	A traffic light cannot turn right
{not Amber, not Brake}	A amber traffic light cannot brake
{not OthTL, not HazLit}	A other traffic light cannot have the hazards light on
{not Red, not IncatLeft}	A red traffic light cannot indicate left
{not Green, not Mov}	A green traffic light cannot move

Table 10 Requirements tables

Requirements	Natural language explanations
{not MedVeh, not Bus}	A medium vehicle cannot be a bus
{not Car, not Xing}	A car cannot be crossing
{not Mobike, not OthTL}	A motorbike cannot be an other traffic light
{not Car, not PushObj}	A car cannot push objects
{not MedVeh, not EmVeh}	A medium vehicle cannot be an emergency one
{not Mobike, not Red}	A motorbike cannot be a red traffic light
{not LarVeh, not Bus}	A large vehicle cannot be a bus
{not Brake, not Cyc}	A cyclist cannot brake
{not MedVeh, not TL}	A traffic light cannot be a medium vehicle
{not Mobike, not Amber}	A motorbike cannot be an amber traffic light
{not LarVeh, not EmVeh}	A large vehicle cannot be an emergency vehicle
{not Mobike, not Green}	A motorbike cannot be a green traffic light
{not MedVeh, not OthTL}	A medium vehicle cannot be an other traffic light
{not HazLit, not Cyc}	A cyclist cannot have the hazard lights on
{not MedVeh, not Red}	A medium vehicle cannot be a red traffic light
{not LarVeh, not TL}	A large vehicle cannot be a traffic light
{not Car, not Ped}	A car cannot be a pedestrian
{not Mobike, not TL}	A motorbike cannot be a traffic light
{not Bus, not EmVeh}	A bus cannot be an emergency vehicle
{not MedVeh, not Amber}	A medium vechile cannot be an amber traffic light
{not LarVeh, not OthTL}	A large vehicle cannot be a other traffic light
{not MedVeh, not Green}	A medium vehicle cannot be a green traffic light
{not Bus, not TL}	A bus cannot be a traffic light
{not LarVeh, not Red}	A large vehicle cannot be a red traffic light
{not Bus, not OthTL}	A bus cannot be a other traffic light
{not LarVeh, notAmber}	A large vehicle cannot be a amber traffic light
{not Cyc, not PushObj}	A cyclist cannot push an object
{not EmVeh, not TL}	An emergency vehicle cannot be a traffic light
{not LarVeh, not Green}	A large vehicle cannot be a green traffic light
{not Bus, not Red}	A bus cannot be a red traffic light
{not EmVeh, not OthTL}	An emergency vehicle cannot be a other traffic light
{not Bus, not Amber}	A bus cannot be an amber traffic light
{not EmVeh, not Red}	A emergency vehicle cannot be a red traffic light
{not Mobike, not Wait2X}	A motorbike cannot wait to cross
{not Bus, not Green}	A bus cannot be a green traffic light
{not TL, not OthTL}	A traffic light cannot be a other traffic light
{not EmVeh, not Amber}	A emergency vehicle cannot be a amber traffic light
{not Mobike, not Xing}	A motorbike cannot be crossing
{not Cyc, not Ped}	A cyclist cannot be a pedestrian
{Ped, Cyc, not Xing}	If an agent is crossing it is either a pedestrian or a cyclist
{not Mobike, not PushObj}	A motorbike cannot push objects
{not EmVeh, not Green}	An emergency vehicle cannot be a green traffic light
{not MedVeh, not Wait2X}	A medium vehicle cannot wait to cross
{not TL, not MovAway}	A traffic light cannot move away

Table 10 (continued)

Requirements	Natural language explanations
{not MedVeh, not Xing}	A medium vehicle cannot be crossing
{not Red, not Amber}	A red traffic light cannot be amber
{not MedVeh, not PushObj}	A medium vehicle cannot push objects
{not MovTow, not TL}	A traffic light cannot move towards
{not OthTL, not MovAway}	A other traffic light cannot move away
{not LarVeh, not Wait2X}	A large vehicle cannot wait to cross
{not TL, not Mov}	A traffic light cannot move
{not Red, not Green}	A red traffic light cannot be green
{not TL, not XingFmRht}	A traffic light cannot be crossing from right
{not Amber, not IncatRht}	An agent cannot indicate right and signal amber
{not Red, not TurLft}	An agent cannot turn left and signal red
{not MovTow, not Mov}	An agent cannot move towards and move
{not TL, not Xing}	A traffic light cannot cross
{not OthTL, not Wait2X}	A other traffic light cannot wait to cross
{not Green, not IncatLeft}	An agent cannot indicate left and signal green
{not Red, not TurRht}	An agent cannot turn right and signal red
{not Amber, not HazLit}	An agent cannot have the hazard lights on and signal amber
{not TL, not PushObj}	A traffic light cannot push objects
{not OthTL, not XingFmLft}	A other traffic light cannot cross from left
{not Green, not IncatRht}	An agent cannot signal green and indicate right
{not Red, not Ovtak}	An agent cannot signal red and overtake
{not Amber, not TurLft}	An agent cannot signal amber and turn left
{not OthTL, not XingFmRht}	A other traffic light cannot cross from right
{not Red, not Wait2X}	An agent cannot signal red and wait to cross
{not Green, not HazLit}	An agent cannot signal green and have the hazard lights on
{not Amber, not TurRht}	An agent cannot signal amber and turn right
{not OthTL, not Xing}	A other traffic light cannot cross
{not Bus, not Ped}	A bus cannot be a pedestrian
{not Red, not XingFmLft}	An agent cannot signal red and cross from left
{not OthTL, not PushObj}	A other traffic light cannot push an object
{not Green, not TurLft}	An agent cannot signal green and turn left
{not Amber, not Ovtak}	An agent cannot signal amber and overtake
{not Mov, not Stop}	An agent cannot move and stop
{not Red, not XingFmRht}	An agent cannot signal red and cross from right
{not Amber, not Wait2X}	An agent cannot signal amber and wait to cross
{not Green, not TurRht}	An agent cannot signal green and turn right
{not Red, not Xing}	An agent cannot signal and cross
{not Amber, not XingFmLft}	An agent cannot signal amber and cross from left
{not Green, not Ovtak}	An agent cannot signal green and overtake
{not Amber, not XingFmRht}	An agent cannot signal amber and cross from right
{not EmVeh, not Ped}	An emergency vehicle cannot be a pedestrian
{not Green, not Wait2X}	An agent cannot signal green and wait to cross
{not Amber, not Xing}	An agent cannot signal amber and cross
{not Green, not XingFmLft}	An agent cannot signal green and cross from left

Table 10 (continued)

Requirements	Natural language explanations
{not Green, not XingFmRht}	An agent cannot signal green and cross from right
{not Green, not Xing}	An agent cannot signal green and cross
{not TL, not Ped}	A traffic light cannot be a pedestrian
{not IncatLeft, not IncatRht}	An agent cannot indicate left and right
{not OthTL, not Ped}	A other traffic light cannot be a pedestrian
{not Brake, not Wait2X}	An agent cannot brake and wait to cross
{not Red, not Ped}	A pedestrian cannot signal red
{not Xing, not Brake}	An agent cannot cross and brake
{not Wait2X, not IncatLeft}	An agent cannot wait to cross and indicate left
{not Brake, not PushObj}	An agent cannot brake and push an object
{not Amber, not Ped}	A pedestrian cannot signal amber
{not Wait2X, not IncatRht}	An agent cannot wait to cross and indicate right
{not Green, not Ped}	A pedestrian cannot signal green
{not Wait2X, not Ovtak}	An agent cannot wait to cross and overtake
{not Wait2X, not XingFmLft}	An agent cannot wait to cross and cross from left
{not Wait2X, not XingFmRht}	An agent cannot wait to cross and cross from right

Table 11 Requirements tables

Requirements	Natural language explanations
{not Wait2X, not Xing}	An agent cannot wait to cross and cross
{not Brake, not Ped}	A pedestrian cannot brake
{not Ped, not IncatLeft}	A pedestrian cannot indicate left
{not Ped, not IncatRht}	A pedestrian cannot indicate right
{not HazLit, not Ped}	A pedestrian cannot have the hazard lights on
{not Cyc, not Green}	A cyclist cannot signal green
{not OutgoLane, not OutgoCycLane}	An outgoing lane cannot be an outgoing cycle lane
{not Ped, not Ovtak}	A pedestrian cannot overtake
{not VehLane, not IncomCycLane}	The vehicle lane cannot be an incoming cycle lane
{not OutgoLane, not IncomCycLane}	An incoming cycle lane cannot be a outgoing lane
{not IncomLane, not OutgoCycLane}	An outgoing cycle lane cannot be an incoming lane
{not OutgoLane, not Pav}	An outgoing lane cannot be a pavement
{not IncomCycLane, not OutgoCycLane}	An incoming cycle lane cannot be an outgoing cycle lane
{not OutgoLane, not RhtPav}	An outgoing lane cannot be a right pavement
{not IncomCycLane, not Pav}	An incoming cycle lane cannot be a pavement
{not XingLoc, not OutgoLane}	A crossing cannot be an outgoing lane
{not VehLane, not Parking}	A vehicle lane cannot be a parking
{not BusStop, not OutgoLane}	A bus stop cannot be a outgoing lane
{not OutgoLane, not Parking}	An outgoing lane cannot be a parking
{not BusStop, not OutgoCycLane}	A bus stop cannot be a outgoing cycle lane
{not Parking, not OutgoCycLane}	A parking cannot be a outgoing cycle lane
{not XingLoc, not IncomCycLane}	A crossing cannot be an incoming cycle lane
{not LftPav, not RhtPav}	A pavement is either on the left or on the right

Table 11 (continued)

Requirements	Natural language explanations
{not IncomLane, not Parking}	An incoming lane cannot be a parking
{not IncomCycLane, not BusStop}	An incoming cycle lane cannot be a bus stop
{not IncomCycLane, not Parking}	An incoming cycle lane cannot be a parking
{not Pav, not Parking}	A pavement cannot be a parking
{not LftPav, not Parking}	A left pavement cannot be a parking
{not RhtPav, not Parking}	A right pavements cannot be a parking
{not Jun, not Parking}	A junction cannot be a parking
{not XingLoc, not BusStop}	A crossing cannot be a bus stop
{not XingLoc, not Parking}	A crossing cannot be a parking
{not BusStop, not Parking}	A bus stop cannot be a parking
{not Mobike, not Ped }	A motorbike cannot be a pedestrian
{not MovTow, not OthTL}	A other traffic light cannot move towards
{not TL, not Brake}	A traffic light cannot brake
{not Red, not MovAway}	A red traffic light cannot move away
{not Amber, not Green}	An amber traffic light cannot be green
{not LarVeh, not Xing}	A large vehicle cannot cross
{not OthTL, not Mov}	A other traffic light cannot move
{not Stop, not TL}	A traffic light cannot stop
{not LarVeh, not PushObj}	A large vehicle cannot push objects
{not Red, not MovTow}	A red traffic light cannot move towards
{not Amber, not MovAway}	A amber traffic light cannot move away
{not Bus, not Wait2X}	A bus cannot wait to cross
{not TL, not IncatLeft}	A traffic light cannot idicate left
{not OthTL, not Brake}	A other traffic light cannot brake
{not Red, not Mov}	A red traffic light cannot move
{not TL, not IncatRht}	A traffic light cannot indicate right
{not Stop, not OthTL}	A other traffic light cannot stop
{not Amber, not MovTow}	A amber traffic light cannot move towards
{not Green, not MovAway}	A green traffic light cannot move away
{not TL, not HazLit}	A traffic light cannot have the hazard lights on
{not Red, not Brake}	A red traffic light cannot brake
{not Bus, not Xing}	A bus cannot be crossing
{not OthTL, not IncatLeft}	A other traffic light cannot indicate left
{not MedVeh, not Ped}	A medium vehicle cannot be a pedestrian
{not Amber, not Mov}	A amber traffic light cannot move
{not Bus, not PushObj}	A bus cannot push objects
{not EmVeh, not Wait2X}	A emergency vehicle cannot wait to cross
{not TL, not Ovtak}	A traffic light cannot overtake
{not Amber, not Stop}	A amber traffic light cannot stop
{not EmVeh, not Xing}	An emergency vehicle cannot be crossing
{not OthTL, not TurLft}	A other traffic light cannot turn left
{not Red, not IncatRht}	A red traffic light cannot indicate right
{not TL, not Wait2X}	A traffic light cannot wait to cross
{not Green, not Brake}	A green traffic light cannot be green and break
{not MovAway, not Mov}	An agentcannot move perpendicularly to and away from the AV

Table 11 (continued)

Requirements	Natural language explanations
{not EmVeh, not PushObj}	An emergency vehicle cannot push an object
{not TurRht, not OthTL}	A other traffic light cannot turn right
{not Amber, not IncatLeft}	An amber traffic light cannot indicate left
{not TL, not XingFmLft}	An traffic light cannot be crossing from the left
{not Red, not HazLit}	A red traffic light cannot have the hazard lights on
{not Green, not Stop}	A green traffic light cannot stop
{not LarVeh, not Ped}	A large vehicle cannot be a pedestrian
{not OthTL, not Ovtak}	A other traffic light cannot overtake

Author contributions EG: conceptualization, methodology, data annotation, software implementation, and writing (original draft preparation); MCS: software implementation, software debugging, and writing; SK: software implementation; FC: supervision, writing reviewing, and editing; TL: supervision, writing reviewing, and editing.

Funding Eleonora Giunchiglia is supported by the EPSRC under the grant EP/N509711/1 and by an Oxford-DeepMind Graduate Scholarship. Mihaela Cătălina Stoian is supported by the EPSRC under the grant EP/T517811/1. Salman Khan is supported by a research agreement with Huawei Technologies Co., Ltd. This work was also supported by the Alan Turing Institute under the EPSRC grant EP/N510129/1, by the AXA Research Fund, by the EPSRC grant EP/R013667/1, and by the European Union's Horizon 2020 research and innovation programme, under grant agreement No. 964505 (E-pi). We also acknowledge the use of the EPSRC-funded Tier 2 facility JADE (EP/P020275/1) and GPU computing support by Scan Computers International Ltd.

Data availability ROAD is publicly available at https://github.com/gurkirt/road-dataset, and the logical constraints will be made publicly available upon publication.

Code availability The code for the SOTA models is available at https://github.com/gurkirt/3D-RetinaNet. The code for the CL, CO, and CLCO models will be made publicly available upon publication.

Declarations

Conflict of interest All authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript.

Ethical approval Not applicable.

Consent to participate Not applicable.

Consent for publication ROAD-R consists of the set of logical constraints in Appendix D, on top of the existing dataset ROAD, which is publicly available, and which is linked. Thus, ethical issues related to person identification do not apply to ROAD-R.

References

Amodei, D., Olah, C., Steinhardt, J., Christiano, P. F., Schulman, J., & Mané, D. (2016). Concrete problems in AI safety. arXiv preprint arXiv:1606.06565

Bailleux, O., & Marquis, P. (2006). Some computational aspects of distance-sat. *Journal of Automated Reasoning, 37*(4), 231–260.

Carreira, J., & Zisserman, A. (2017). Quo vadis, action recognition? A new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.*.

Curtis, K., Awad, G., Rajput, S., & Soboroff, I. (2020). HLVU: A new challenge to test deep understanding of movies the way humans do. In *Proceedings of the 2020 International Conference on Multimedia Retrieval*.

Demsar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research, 7*, 1–30.

Diligenti, M., Gori, M., & Sacca, C. (2017). Semantic-based regularization for learning and inference. *Artificial Intelligence, 244*, 143–165.

Diligenti, M., Roychowdhury, S., & Gori, M. (2017). Integrating prior knowledge into deep learning. In *2017 16th IEEE international conference on machine learning and applications (ICMLA)*.

Donadello, I., Serafini, L., & d'Avila Garcez, A. (2017). Logic tensor networks for semantic image interpretation. In *Proceedings of IJCAI*.

Dragone, P., Teso, S., & Passerini, A. (2021). Neuro-symbolic constraint programming for structured prediction. In *IJCLR-NeSy*.

Eén, N., & Sörensson, N. (2004). An Extensible SAT-solver. In E. Giunchiglia and A. Tacchella (Eds.), *Theory and applications of satisfiability testing*. SAT 2003. Lecture Notes in Computer Science, vol. 2919. Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-540-24605-3_37

Feichtenhofer, C., Fan, H., Malik, J., & He, K. (2019). SlowFast networks for video recognition. In *Proceedings of the IEEE/CVF international conference on computer vision*.

Giunchiglia, E., & Lukasiewicz, T. (2020). Coherent hierarchical multi-label classification networks. *Advances in Neural Information Processing Systems, 33*(2020), 9662–9673.

Giunchiglia, E., & Lukasiewicz, T. (2021). Multi-label classification neural networks with hard logical constraints. *Journal of Artificial Intelligence Research, 72*(2021), 759–818.

Giunchiglia, E., Stoian, M. C., & Lukasiewicz, T. (2022). Deep learning with logical constraints. In *Proceedings of IJCAI*.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*

Hickey, R., & Bacchus, F. (2019). Speeding up assumption-based SAT. In *Proceedings of SAT*.

Hoernle, N., Karampatsis, R., Belle, V., & Gal, K. (2022). MultiplexNet: Towards fully satisfied logical constraints in neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Hu, Z., Ma, X., Liu, Z., Hovy, E., & Xing, E. (2016). Harnessing deep neural networks with logic rules. arXiv preprint arXiv:1603.06318 (2016)

Hu, Z., Yang, Z., Salakhutdinov, R., & Xing, E. (2016). Deep neural networks with massive learned knowledge. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Hua, Y., Zhao, Z., Liu, Z., Chen, X., Li, R., & Zhang, H. (2018). Traffic prediction based on random connectivity in deep learning with long short-term memory. In *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall). IEEE.*

Hájek, P. (1998). *Metamathematics of Fuzzy Logic*. Dordrecht, the Netherlands: Kluwer Academic Publishers.

Kalogeiton, V., Weinzaepfel, P., Ferrari, V., & Schmid, C. (2017). Action tubelet detector for spatio-temporal action localization. In *Proceedings of the IEEE International Conference on Computer Vision*.

LeCun, Y., Bottou, L., Orr, G., & Müller, K. (2012). Efficient backprop. In *Neural Networks: Tricks of the Trade*. Springer, Heidelberg.

Li, C. M., & Manyà, F. (2009). Maxsat, hard and soft constraints. In *Handbook of Satisfiability. Frontiers in Artif. Intell. and Appl.,* 185(2009), 613-631.

Li, T., & Srikumar, V. (2019). Augmenting neural networks with first-order logic. arXiv preprint arXiv:1906.06298 (2019)

Li, D., Qiu, Z., Dai, Q., Yao, T., & Mei, T. (2018). Recurrent tubelet proposal and recognition networks for action detection. In *Proceedings of the European conference on computer vision*.

Maddern, W., Pascoe, G., Linegar, C., & Newman, P. (2017). 1 year, 1000 km: The Oxford RobotCar dataset. *The International Journal of Robotics Research, 36*(1), 3–15.

Mahon, L., Giunchiglia, E., Li, B., & Lukasiewicz, T. (2020). Knowledge graph extraction from videos. In *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*.

Schietgat, L., Vens, C., Struyf, J., Blockeel, H., Kocev, D., & Dzeroski, S. (2010). Predicting gene function using hierarchical multi-label decision tree ensembles. *BMC Bioinformatics, 11*, 1–14.

Singh, G., Akrigg, S., Maio, M. D., Fontana, V., Alitappeh, R. J., Saha, S., Saravi, K. J., Yousefi, F., Culley, J., Nicholson, T., Omokeowa, J., Khan, S., Grazioso, S., Bradley, A., Gironimo, G. D., & Cuzzolin, F. (2022). ROAD: The road event awareness dataset for autonomous driving. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 45*(1), 1036–1054.

Singh, G., & Cuzzolin, F. (2019). Recurrent convolutions for causal 3D CNNs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*.

Sommerville, I. (2011). *Software Engineering*. London: Pearson.

Vens, C., Struyf, J., Schietgat, L., Dzeroski, S., & Blockeel, H. (2008). Decision trees for hierarchical multi-label classification. *Machine Learning, 73*(2008), 185–214.

Wang, X., Girshick, R., Gupta, A., & He, K. (2018). Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Wehrmann, J., Cerri, R., & Barros, R. C. (2018). Hierarchical multi-label classification networks. In *International Conference on Machine Learning Proceedings, ICML.*

Xu, J., Zhang, Z., Friedman, T., Liang, Y., & Van den Broeck, G. (2018). A semantic loss function for deep learning with symbolic knowledge. In *Proceedings of International Conference on Machine Learning*.