POLITECNICO DI TORINO
Repository ISTITUZIONALE

L-TReiD: Logic Tensor Transformer for Re-Identification

Original
L-TReiD: Logic Tensor Transformer for Re-Identification / Russo, A., Manigrasso, F., Lamberti, F., Morra, L.. - STAMPA. - (2023), pp. 345-357. (International Symposium on Visual Computing (ISVC) 2023 Lake Tahoe, Nevada, USA 16/10/2023 - 18/10/2023) [10.1007/978-3-031-47966-3_27].

Availability:
This version is available at: 11583/2982375 since: 2023-12-19T11:43:14Z

Publisher:
SPRINGER

Published
DOI:10.1007/978-3-031-47966-3_27

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
Springer postprint/Author's Accepted Manuscript

This version of the article has been accepted for publication, after peer review (when applicable) and is subject to Springer Nature's AM terms of use, but is not the Version of Record and does not reflect post-acceptance improvements, or any corrections. The Version of Record is available online at: http://dx.doi.org/10.1007/978-3-031-47966-3_27

(Article begins on next page)

13 October 2026

L-TReiD: Logic Tensor Transformer for Re-Identification

Russo Alessandro[1 [0009−0009−6711−8277]], Manigrasso Francesco[1 [0000−0002−4151−8880]], Lamberti Fabrizio[1 [0000−0001−7703−1372]], and Morra Lia[1 [0000−0003−2122−7178]]

Politecnico di Torino, Torino TO 10129, Italy
{alessandro.russo, francesco.manigrasso, fabrizio.lamberti, lia.morra}@polito.it

Abstract. This article proposes a Neuro-Symbolic (NeSy) machine learning approach to Object Re-identification. NeSy is an emerging branch of artificial intelligence which combines symbolic reasoning and logic-based knowledge representation with the learning capabilities of neural networks. Since object re-identification involves assigning the identity of the same object across different images and different conditions, such a task could benefit greatly from leveraging the logic capabilities of a NeSy framework to inject prior knowledge about invariant properties of the objects. To test this assertion, we combined the Logic Tensor Networks (LTNs) NeSy framework with a state-of-the-art Transformer-based Re-Identification and Damage Detection Network (TransRe3ID). The LTN incorporates prior knowledge about the properties that two instances of the same object have in common. Experimental results on the Bent&Broken Bicycle re-identification dataset demonstrate the potential of LTNs to improve re-identification systems and provide novel opportunities to identify pitfalls during training.

Keywords: Neuro-Symbolic learning · Logic Tensor Networks · Object Re-Identification · Transformers.

1 Introduction

In recent years, the novel field of Neuro-Symbolic Machine Learning (NeSy) has emerged at the intersection of symbolic artificial intelligence and neural networks [7, 8, 10]. NeSy combines the strengths of both symbolic reasoning and deep learning to address complex problems by combining the expressive power of logic-based knowledge representation with the learning capabilities of neural networks. While NeSy techniques have been applied in a variety of visual tasks including scene graph extraction [3, 4, 12], object detection [17], zero-shot image classification and recognition [19, 25] and visual query answering [18, 20, 23], to the best of our knowledge ours is the first attempt to exploit them in the context of Object Re-Identification (ReID).

Current re-identification methods simply perform implicit pattern matching while not making use of more readily available information about similar objects

like color and object type labels. The proposed approach could prove beneficial in this sense by using logical rules that encode prior knowledge about properties that are invariant across different images of the same object instance. Specifically, we focus on improving the baseline performance of the transformer-based re-identification and damage detection network (TransRe3ID) [21] by incorporating Logic Tensor Networks (LTNs) [1].

To evaluate our approach, we conduct experiments with the synthetic Bent & Broken bicycle Damage Detection and Re-Identification dataset [21]. In this benchmark, the goal is to re-identify the same object (in this case, a bicycle) in the presence of damages and missing parts, and hence the network must distinguish between large inter-object deformations (e.g., induced by an incident) and subtle intra-object differences (e.g., due the difference texture of the bike frame). In addition, the dataset includes challenges commonly associated to ReID such as changes in viewpoint, and the presence of dirt.

The remainder of this paper is organized as follows. Section 2 provides a brief background on the Logic Tensor Networks framework and an analysis of recent related work in the field of object ReID, with a particular focus on the use of transformers in this area. Section 3 outlines the methodology with the definition of the FOL rules, their integration into the TransRe3ID network, and the loss formulation. Section 4 presents the experimental setup, including the dataset and evaluation metrics. Section 5 discusses the results and analyzes the impact of integrating LTNs on the performance of the ReID system. It also examines the extent to which the proposed logical rules were satisfied during training to explain the properties of the resulting network. Finally, Section 6 concludes the paper and highlights possible directions for future research in this area.

2 Background

2.1 Object Re-Identification

Object ReID is the task of identifying the same object in multiple images, regardless of its position, illumination, or context. The use of transformers has rapidly increased in recent years, with applications ranging from person ReID [16, 24, 27] to vehicle ReID [14, 15, 26]. One of the best representatives of these works is TransReID proposed by [9], which presents a ViT backbone followed by two separate ReID branches, one based on global attention and one that enforces local attention by using a separate module that extracts random local parts of the image. This work was extended by [21] to perform simultaneous damage detection and object ReID, resulting in the TransReI3D architecture trained on the Bent&Broken Bicycle ReID dataset.

2.2 Logic Tensor Networks

Logic Tensor Networks (LTNs), originally proposed by d'Avila Garcez and Serafini [6, 1, 2], have been used in a variety of different tasks, from object recogni-

tion [17, 22, 4] to reinforcement learning [2] and sentiment analysis [11], demonstrating their flexibility as a NeSy framework. They combine deep neural networks with a first-order logical knowledge representation. In short, LTNs use a fuzzy logical language, called real-world logic, as the underlying formalism, which consists of a First-Order Logic language (FOL) whose signature includes constants, function, and predicate symbols. Since there is no complete certainty in real-world problems and formulas may be partially true, fuzzy semantics is used as an approximation to real logic, using the concept of *grounding* to define how symbols are concretely interpreted by tensors in the real field.

Given a vector space R^n and a set of predicates $\mathcal{P}$, a grounding $\mathcal{G}$ has the following properties:

$$\mathcal{G}(P) \in R^{n \star k} \rightarrow [0,1], \forall p \in \mathcal{P}$$

As such, predicate symbols are interpreted as functions mapping real vectors to the $[0, 1]$ interval, which can be interpreted as the truth level of the predicate. A typical example is the unary predicate *is-a*, which determines the existence of a certain object or property associated with it. Consider the following use case as an example: if $b = \mathcal{G}(x)$ is the grounding for the image of a bicycle, then $\mathcal{G}(Bike)(b) \simeq 1$. A logical condition expressed in FOL allows to define its properties, e.g. $\forall x(\text{Bike}(x) \rightarrow \text{hasWheels}(x))$. Thus, the truth value of a logical condition can be calculated by a neural network by first computing the grounding of the unary symbols contained in the logical clause and then combining them through fuzzy logical operators and quantifiers.

Fuzzy logic formulas can be associated with fuzzy logic operators such as conjunctions ($\wedge$), disjunctions ($\vee$), and implications ($\implies$), including logical quantifiers ($\forall$ and $\exists$). Several real-valued differentiable implementations are available in the fuzzy logic domain [13]. The implementation here used follows the one in [1], which is based on the Lukasiewicz formulation [5]:

$$\begin{aligned} \mathcal{G}(-\phi) &= 1 - \mathcal{G}(\phi) \\ \mathcal{G}(\sigma \vee \psi) &= \min(1, \mathcal{G}(\phi) + \mathcal{G}(\psi)) \end{aligned} \tag{1}$$

The combination of connectors, predicates and quantifiers defines axioms, for which examples can be found in Section 3.2.

The set of closed formulas such as axioms and logical labels is called a knowledge base $\mathcal{K}$, which stands in combination with the grounding on our examples. In practice, such a grounding is only partially defined for optimization purposes, since our set $\mathcal{K}$ is qualitatively a finite and limited set of examples.

Best Satisfiability Problem Given a grounding $\hat{\mathcal{G}}_\theta$, where θ is the set of parameters of all predicates, the learning problem in LTNs is formulated as a best satisfiability problem, where the goal is to determine the values of Θ^* that maximize the truth values of the conjunction of all closed formulas $\phi \in \mathcal{K}$:

$$\Theta^* = \text{argmax}_\Theta \, \hat{\mathcal{G}}_\theta \left(\bigwedge_{\phi \in \mathcal{K}} \phi \right) - \lambda \|\Theta\|_2^2 \tag{2}$$

where $\lambda\|\Theta\|_2^2$ is a regularization term. In real world cases, fully satisfying a grounded theory is highly unlikely given the possibility of exceptions to each rule. Thus, we opt to instead find the grounding that obtains the highest satisfaction while taking into account such exceptions. Examples of these exceptions are common in the visual domain, from occasional deviations from the norm, to features that may not be always visible. For example, a bicycle (normally) has both wheels, but a damaged bicycle or a bicycle under repair may have one or both wheels missing or simply obscured.

3 Methodology

3.1 Overall Architecture

Following the work of [21], we use the TrainsReI3D architecture as backbone, which was already demonstrated on the selected benchmark for damaged object ReID (more information about the dataset can be found in Section 4.1). This architecture performs multi-task damage detection and ReID by using a ViT backbone and having three output branches: one for classifying the presence of damage and missing parts, one that performs object ReID on the whole image and the last one, called Jigsaw Patch Branch, that performs object ReID based on local features. This last branch works by using the Jigsaw Patch Module (JPM), which takes as input the patch tokens and randomly reshuffles them into four separate subsets of equal size, each containing a copy of the original [cls] patch token. For more details on the underlying transformer network, the reader can refer to the works of [9, 21]. Here we have also added a separate output branch for predicting additional attributes of each bike instance, with a separate classification head for each attribute (multi-attribute, multi-class classification).

Figure 1 shows a representation of the complete architecture.

3.2 Logic Tensor Network Definition

Our LTN-based NeSy approach defines predicates on the images, that are in turn grounded by the output [cls] token features of the ReID and auxiliary branches to obtain subsymbolic representations of IDs (x_{reid}) and bike attributes(x_{attr}), respectively. The core of the LTN is the definition of the knowledge base , which combines known facts (in the present case, labeled instances) with logical constraints. The Bent&Broken Bicycle dataset, as defined in Section 4.1, defines each unique bicycle ID based on the unique combination of instance attributes, and at the same time, contains damaged and undamaged versions of the same bicycle ID. The LTN was used to inject into the learning process prior knowledge that directly derives from the rules underlying the dataset labeling structure. Specifically, we make three basic observations from which we define our knowledge base rules: (1) “Images with the same auxiliary attributes must have the same ID”; (2) “Images of the same object, but with different damage types, must have the same ID”; (3) “Images with the same ID must be associated with

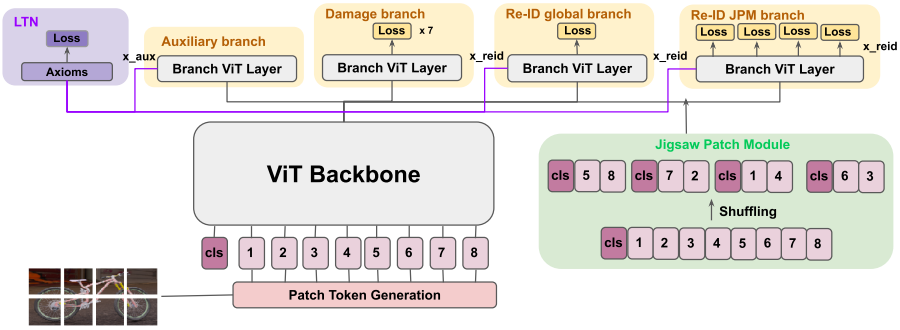


Fig. 1: Architecture of the proposed L-TransReI3D. The original architecture of TransReI3D [21] consists of a ViT backbone that feeds into 3 output branches: (1) the Damage Branch to classify the presence of damage and missing parts; (2) the global ReID branch for the ReID of objects in the global image; and (3) the ReID JPM branch for the local ReID of objects in subsets of image patches as an additional self-supervised task. The architecture has been further extended with an Auxiliary branch, which performs the classification of bike attributes. With the exception of the Damage branch, each branch also feeds into the LTN axioms for the LTN loss calculation.

feature vectors that are close in the embedding space". In the rest of this section, we define in detail the variables and predicates that form the FOL, and the axioms encoding prior knowledge.

Predicates The proposed LTN is based on three predicates:

- **IDp**(x_{reid}, ID) is a trainable ID predicate classifier, where x_{reid} is an input image and ID is a term denoting a ID variable. The predicate should return the probability that an image x_{reid} belongs to ID. The predicate is grounded by a classification layer followed by a softmax function, where x_{reid} is grounded by the output [cls] token feature extracted from the ReID branch;
- **AUXp**(x_{aux}, AUX_i) is a trainable predicate classifier for bike attributes, where x_{aux} is the image and AUX_i is the corresponding attribute label (more information in Section 4.1). Each $AUXp$ returns the probability that an image x_{aux} belongs to the label of the bike attributes AUX_i. Each predicate is grounded by a classification layer followed by a softmax function, where x_{aux} is grounded by the output [cls] feature extracted from the auxiliary branch;
- **SameInstance**(x_{reid_i}, x_{reid_j}) is a non-trainable predicate indicating whether two images x_{reid_i} and x_{reid_j} contain bikes belonging to the same ID; the predicate is grounded by the cosine similarity between the two input features followed by a sigmoid function, whereas x_{reid_i} and x_{reid_j} are grounded

by the output class token features of each image extracted from the ReID branch.

Fuzzy Operators Here are defined the fuzzy logic operators introduced in the LTN formalism and used in the axioms:

- **Diagonal Quantification** $\text{Diag}(x, \ldots, l)$ quantifies over tuples combining the i-th instance of each of the variables in the argument of Diag [1]. For example, for a dataset with samples x and target label y, $\forall\text{Diag}(x, y)$ quantifies over each (sample, label) pair.
- **Equivalence Connection** $Pred_1 \Leftrightarrow Pred_2$ corresponds to the equivalence logic operator, which states that the two predicates should have the same truth value.
- **Guarded Quantification** $Quantifier(Condition\ Variables : Condition\ Mask\{Maskedpredicates\})$ applies a mask to select terms to be included in the quantification based on a selected condition variables. For example, given a binary predicate P applied to variables x and y, the axiom $\forall[x, y], x, y : x == y\{P(x, y)\}$ computes only any combination of x and y for which the condition $x == y$ is satisfied.

Axioms

The following rules enforce the three observations defined above in dual pairs of positive-negative axioms:

- **Rule #1: Images with the same auxiliary attributes must be assigned to the same ID.** After selecting only pairs of images that (do not) have the same ID (guarded condition), the rules enforce the constraint that the auxiliary attributes of images x_{aux_i} and x_{aux_j} are the same if and only if the two corresponding images x_{reid_i} and x_{reid_j} have the same IDs (and vice versa). The constraint is applied for each auxiliary attribute:

$$\begin{aligned} &\forall\text{Diag}(x_{reid_i}, ID_i, x_{aux_i}, AUX_{k_i}) \left(\forall\text{Diag}(x_{reid_j}, ID_j, x_{aux_j}, AUX_{k_j})\right. \\ &\mathbf{ID_i}, \mathbf{ID_j} : \mathbf{ID_i} == \mathbf{ID_j} \left\{\left((\text{AUXp}(x_{aux_i}, AUX_{k_i}) \Leftrightarrow \text{AUXp}(x_{aux_j}, AUX_{k_j})\right) \Leftrightarrow\right. \\ &\left.\left.\left(\text{IDp}(x_{reid_i}, ID_i) \Leftrightarrow \text{IDp}(x_{reid_j}, ID_j)\right)\right\}\right) \end{aligned} \tag{3}$$

$$\begin{aligned} &\forall\text{Diag}(x_{reid_i}, ID_i, x_{aux_i}, AUX_{k_i}) \left(\forall\text{Diag}(x_{reid_j}, ID_j, x_{aux_j}, AUX_{k_j})\right. \\ &\mathbf{ID_i}, \mathbf{ID_j} : \mathbf{ID_i}! = \mathbf{ID_j} \left\{\neg\left((\text{AUXp}(x_{aux_i}, AUX_{k_i}) \Leftrightarrow \text{AUXp}(x_{aux_j}, AUX_{k_j})\right) \Leftrightarrow\right. \\ &\left.\left.\neg\left(\text{IDp}(x_{reid_i}, ID_i) \Leftrightarrow \text{IDp}(x_{reid_j}, ID_j)\right)\right\}\right) \end{aligned} \tag{4}$$

– **Rule #2: Images with different damage types, but with the same ID label, must be assigned to the same ID.** After selecting only pairs of images that (do not) have the same ID (guarded condition), we ask the network to ensure that two images, one with an undamaged instance $x_{reid_{und_i}}$ and one with a damaged instance $x_{reid_{dmg_j}}$, (do not) have the same IDs; this is computed for each type of damage and each missing part

$$\begin{aligned} &\forall \text{Diag}(x_{reid_{dmg_i}}, ID_i) \ \Big(\forall \text{Diag}(x_{reid_{und_j}}, ID_j) \\ &\mathbf{ID_i}, \mathbf{ID_j} : \mathbf{ID_i} == \mathbf{ID_j} \ \Big\{ \text{IDp}(x_{reid_{dmg_i}}, ID_i) \Leftrightarrow \text{IDp}(x_{reid_{und_j}}, ID_j) \Big\} \Big) \end{aligned} \tag{5}$$

$$\begin{aligned} &\forall \text{Diag}(x_{reid_{dmg_i}}, ID_i) \ \Big(\forall \text{Diag}(x_{reid_{und_j}}, ID_j) \\ &\mathbf{ID_i}, \mathbf{ID_j} : \mathbf{ID_i}! = \mathbf{ID_j} \ \Big\{ \neg \Big(\text{IDp}(x_{reid_{dmg_i}}, ID_i) \Leftrightarrow \text{IDp}(x_{reid_{und_j}}, ID_j) \Big) \Big\} \Big) \end{aligned} \tag{6}$$

– **Rule #3: Images with the same ID labels must be grounded by feature vectors that are close in the embedding space.** After selecting only pairs of images that (do not) have the same ID (guarded condition), we ask the network to ensure that two images (do not) have close output features from the ReID branch

$$\begin{aligned} &\forall \text{Diag}(x_{reid_i}, ID_i) \ \Big(\forall \text{Diag}(x_{reid_j}, ID_j) \\ &\mathbf{ID_i}, \mathbf{ID_j} : \mathbf{ID_i} == \mathbf{ID_j} \Big\{ \text{SameInstance}(x_{reid_i}, x_{reid_j}) \Big\} \Big) \end{aligned} \tag{7}$$

$$\begin{aligned} &\forall \text{Diag}(x_{reid_i}, ID_i) \ \Big(\forall \text{Diag}(x_{reid_j}, ID_j) \\ &\mathbf{ID_i}, \mathbf{ID_j} : \mathbf{ID_i}! = \mathbf{ID_j} \Big\{ \neg \text{SameInstance}(x_{reid_i}, x_{reid_j}) \Big\} \Big) \end{aligned} \tag{8}$$

We also add another set of axioms for the classification of the k-th bike attribute:

$$\forall \text{Diag}(x_{aux}, AUX_k) \ \text{AUXp}(x_{aux}, AUX_k) \tag{9}$$

In addition to the axioms above, we also analyze an additional axiom which computes the ID classification as follows:

$$\forall \text{Diag}(x_{reid}, ID) \ \text{IDp}(x_{reid}, ID) \tag{10}$$

This axiom is not included in the total satisfiability computation (that is, the LTN loss) since the ID classification is already trained through the global ReID branch.

Each axiom is computed five times, once for the output of the global ReID branch and once for each of the four local outputs of the JPM branch.

3.3 Loss Computation

For the loss calculation, after computing $TotSat$ as the final satisfiability of the Knowledge Base, we add the obtained LTN loss as $\mathcal{L}_{LTN} = 1 - TotSat$ to the original loss of TransReI3D $\mathcal{L}_{TReI3D}$, which has the following form:

$$\mathcal{L}_{final} = \mathcal{L}_{TReI3D} + \mathcal{L}_{LTN} \tag{11}$$

with $\mathcal{L}_{TReI3D}$ having the following structure from [21]:

$$\begin{aligned} \mathcal{L}_{TReI3D} = \alpha\mathcal{L}_{ID}\left(f_g\right) + \beta\mathcal{L}_{T}\left(f_g\right) + \gamma\mathcal{L}_{\mathcal{D}}\left(f_d\right) \\ + \frac{1}{k}\sum_{j=1}^{k}\left(\mathcal{L}_{ID}\left(f_l^j\right) + \mathcal{L}_{T}\left(f_l^j\right)\right) \end{aligned} \tag{12}$$

where $\mathcal{L}_T$ and $\mathcal{L}_{ID}$ are the triplet loss and the ID cross-entropy loss, $\mathcal{L}_D$ is the damage detection loss, k (= 4) is the number of classification outputs of the JPM branch, and f_g, f_l, and f_d are the output [cls] features of the global branch, the jigsaw branch, and of the damage detection branch, respectively. To compute $\mathcal{L}_T$, triplets with hard negative and positive mining are sampled online from each batch.

At inference time, the logical rules are not used, as they are only used to support model training by integrating a semantic component to the loss.

4 Experimental Settings

4.1 Dataset

We used the Bent&Broken bicycle dataset [21] with the same split for training and validation as in the original paper. The dataset contains synthetically generated bicycle images both before and after undergoing damage to simultaneously solve the Damage Detection and ReID tasks, and includes challenges such as missing parts, changes in viewpoint, and dirt. The dataset also contains additional attributes for each image, such as the model, type of bicycle, color and texture of the frame, and presence of stickers. The dataset contains a total of 39,200 images of 2,800 unique IDs, each defined by a unique combination of additional information. There are two types of damage, namely frame bending and frame breaking, which are not mutually exclusive, and up to five parts can be missing from each bicycle, including the two wheels, hand brake, pedals and seat. There are 20 base models, which are divided into 6 types of bicycles. For the frame, there are six texture patterns in different styles, with both the base and pattern colors selected from a pool of 50 colors each. A bike can contain a top and/or bottom sticker, both selected from a pool of 11 stickers. We performed two sets of experiments: one in which only the bike type and model are predicted, and one in which all attributes are used. Based on preliminary experiments, bike

type and model appeared potentially easier to classify, also taking into account that random data augmentation may occasionally disturb the detection of color, texture and stickers.

4.2 Hyperparamenters

Experiments were performed on a NVIDIA TITAN XP with 12 GB of VRAM. L-TransReI3D was trained for 20 epochs with batch size equal to 16, learning rate 1e-2, weight decay 1e-4, and OneCycle scheduler with a warm-up period of 5 epochs. The experiments were run 3 times with and without the LTN component. The metrics used were mean average precision (mAP) and Recall@K with K=1, 5, 10.
Each trainable predicate consists of a FC layer with an input size of 768, equal to the embedding size, and an output size that depends on the predicted information (i.e., ID, bike model, bike type, etc..). The total satisfiability of the knowledge base $TotSat$ was calculated in the following order: first, for each dual rule pair defined in Section 3.2, the axioms calculated on the global ReID branch output and the local ReID JPM branch outputs are aggregated, obtaining their respective rule satisfiability; then all rule satisfiabilities are aggregated to obtain the total final satisfiability.

For the aggregation, as suggested by [1], we approximate the universal $\forall$ quantifier with the generalized mean w.r.t. error aggregator:

$$A_{pME} = 1 - \left(\frac{1}{n} \sum_{i=1}^{n} (1 - a_i)^{p_\forall} \right)^{\frac{1}{p_\forall}} \tag{13}$$

with the parameter $p_\forall$ defining the stringency in computing the satisfiability of logical constraints. This parameter was scheduled starting at $p_\forall = 2$ and then increasing to $p_\forall = 4$ at the halfway point, after the 10th epoch, since higher p values at the beginning of training might prevent the network from converging, as previously reported by [1]. It should be noted that, while the A_{pME} is also used to aggregate all axioms (see Eq. 2), the scheduling of $p_\forall$ was only applied to the $\forall$ quantifiers.

5 Results

The results comparison of L-TReI3D with the original TransReI3D architecture are shown in Table 1. Two versions of the LTN were compared, using a subset (mAP=85.7) or all (mAP=85.0) the attributes, and both achieve similar performance to the original network (mAP=85.3). Fig. 2 shows some query examples with the corresponding similarity values.

Information about the learning behavior of the network can also be extracted from the satisfiability of the logical rules. For this purpose, Figure 3 shows, for each rule presented in Section 3.2, the satisfiability of their respective dual axiom

Fig. 2: Retrieval results (Top-5 images) for the network (ID and similarity scores). The correct ID is retrieved despite the presence of missing parts (ID 15), bent (ID 15), or broken (ID 19) frame.

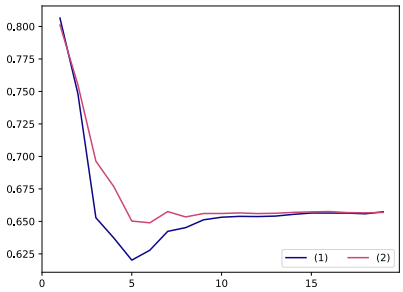
Table 1: Performance comparison of L-TransReI3D with the original TransReI3D on the validation set. Mean average accuracy (mAP) and R@K with K=1,5,10 are given along with standard deviation over 3 replicates. L-TReID (subset) represents the configuration using only a subset of the additional features, while L-TReID (all) uses all of them.

Network	mAP	R@1	R@5	R@10
TransReI3D	85.3 ± 0.2	79.8 ± 0.7	91.9 ± 1.1	96.3 ± 0.5
L-TReI3D (subset)	**85.7 ± 0.1**	79.8 ± 0.5	**93 ± 0.3**	96.2 ± 0.4
L-TReI3D (all)	85.0	79.1	93.1	96.3

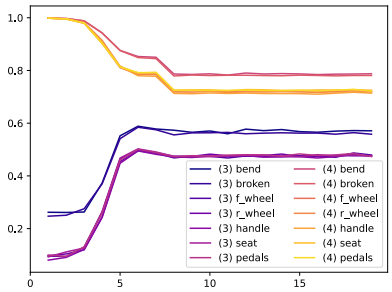
pairs; results were reported on the experiment in which all attributes were used during training.

As shown in Figure 3a, for Rule #1, axioms (3) and (4) start with a satisfiability of 0.82 and then drop to 0.66 each: this, together with the satisfiability of (8) and (9) and the classification accuracies obtained on the additional bike attributes (which are lower than 20%), suggests that the network has difficulty classifying the bike attributes and correlating them with the corresponding IDs. This could also indicate that the selected bike attributes are not as relevant to the network during ID classification as the training progresses.

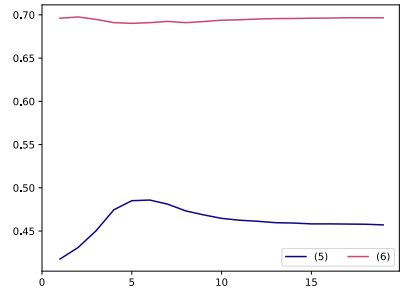
In Figure 3b, for Rule #2, axioms (5) and (6) - shown separately for each individual type of predicted damage and missing part - provide another interesting clue: while the performance of the positive axiom (5) slowly decreases as training progresses, the performance of negative axiom (6), on the contrary, increases. This suggests that, as training progresses, the network finds it more difficult to associate damaged and undamaged instances of the same bike to the same IDs. In addition, axioms related to bent and broken damage show stronger behavior than those related to missing parts, suggesting that the former have a stronger influence on the classification. An explanation for this last result could



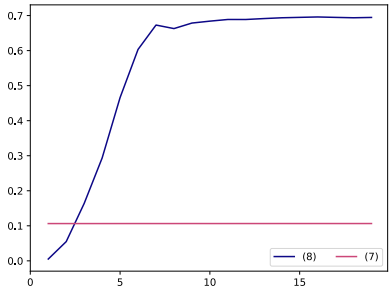
(a) Axioms (3) and (4) for Rule #1



(b) Axioms (5) and (6) for Rule #2



(c) Axioms (7) and (8) for Rule #3



(d) Axioms (9) and (10)

Fig. 3: Satisfiability for the proposed axioms over 20 epochs. Axioms (5) and (6) are depicted separately for each individual type of predicted damage and missing part. Axioms (9) and (10) (bottom right) are also shown together because of the similarity of their logical rules.

likely be derived from the fact that most of the auxiliary attributes that define an ID in the Bent&Broken Bicycle dataset (see Section 4.1), such as bicycle type, model, sticker, and frame color and texture, are associated with the frame and not with the missing parts, making the changes to the frame caused by the two types of damage more influential on the ReID task than the removal of any of the other parts.

In Figure 3c, for Rule #3, axioms (7) and (8) provide some information about the behavior of the network when it relates features of frames with similar and different IDs: the positive axiom (7) indicates that the network is relatively good at associating features of bikes belonging to the same IDs from the beginning, while the negative axiom (8) shows that it has difficulty separating features of bikes with different IDs.

Finally, in Figure 3d, Axiom (9) shows that the network has difficulty classifying bike attributes, which may suggest that such fine-grained characteristics may be too difficult to learn using only an LTN-based loss. Second, Axiom (10) shows that the network only partially learns to classify IDs correctly, suggesting that the effectiveness of the training process largely depends on the triplet loss, rather than the cross entropy used for ID classification.

6 Conclusions

This research investigated the application of NeSy machine learning to object ReID by combining the LTN framework with the TransRe3ID network baseline. Experimental results show the feasibility of applying this framework on a complex computer vision task such as instance-level image retrieval. On the other hand, we observed limited performance improvements, probably due to the fact that the network was able to recover from the Bent&Broken dataset from those already implicit in the original ID labels. Classifying the auxiliary attributes has also proven to be a difficult task for the network. On the other hand, the integration of NeSy machine learning techniques provides a novel method for analyzing the training behavior of the network, which could potentially pave the way for more effective explicability of object ReID architectures. Future research may explore NeSy techniques that differ from LTNs, evolve the proposed logical constraints by providing additional or entirely new rules, or, as mentioned earlier, attempt to optimize and extend the classification task for bike attributes. Finally, the proposed technique should be further investigated in the small data regime.

Acknowledgements

This study was carried out within the FAIR - Future Artificial Intelligence Research and received funding from the European Union Next-GenerationEU (PIANO NAZIONALE DI RIPRESA E RESILIENZA (PNRR) – MISSIONE 4 COMPONENTE 2, INVESTIMENTO 1.3 – D.D. 1555 11/10/2022, PE00000013). This manuscript reflects only the authors' views and opinions, neither the European Union nor the European Commission can be considered responsible for them.

References

1. Badreddine, S., Garcez, A.d., Serafini, L., Spranger, M.: Logic tensor networks. Artificial Intelligence **303**, 103649 (2022)
2. Badreddine, S., Spranger, M.: Injecting prior knowledge for transfer learning into reinforcement learning algorithms using logic tensor networks. arXiv preprint arXiv:1906.06576 (2019)

3. Chen, B., Marussy, K., Pilarski, S., Semeráth, O., Varro, D.: Consistent scene graph generation by constraint optimization. In: Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering. pp. 1–13 (2022)
4. Donadello, I., Serafini, L.: Compensating supervision incompleteness with prior knowledge in semantic image interpretation. In: 2019 International Joint Conference on Neural Networks (IJCNN). pp. 1–8. IEEE (2019)
5. Dutta, S., Basu, S., Chakraborty, M.K.: Many-valued logics, fuzzy logics and graded consequence: a comparative appraisal. In: Logic and Its Applications: 5th Indian Conference, ICLA 2013, Chennai, India, January 10-12, 2013. Proceedings 5. pp. 197–209. Springer (2013)
6. Garcez, A.d., Gori, M., Lamb, L.C., Serafini, L., Spranger, M., Tran, S.N.: Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning. arXiv preprint arXiv:1905.06088 (2019)
7. Garcez, A.d., Lamb, L.C.: Neurosymbolic ai: The 3 rd wave. Artificial Intelligence Review pp. 1–20 (2023)
8. Gibaut, W., Pereira, L., Grassiotto, F., Osorio, A., Gadioli, E., Munoz, A., Gomes, S., dos Santos, C.: Neurosymbolic ai and its taxonomy: a survey. arXiv e-prints pp. arXiv–2305 (2023)
9. He, S., Luo, H., Wang, P., Wang, F., Li, H., Jiang, W.: Transreid: Transformer-based object re-identification. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 15013–15022 (2021)
10. Hitzler, P.: Neuro-symbolic artificial intelligence: The state of the art (2022)
11. Huang, H., Zhang, B., Jing, L., Fu, X., Chen, X., Shi, J.: Logic tensor network with massive learned knowledge for aspect-based sentiment analysis. Knowledge-Based Systems **257**, 109943 (2022)
12. Khan, M.J., Breslin, J.G., Curry, E.: Expressive scene graph generation using commonsense knowledge infusion for visual understanding and reasoning. In: European Semantic Web Conference. pp. 93–112. Springer (2022)
13. van Krieken, E., Acar, E., van Harmelen, F.: Analyzing differentiable fuzzy logic operators. Artificial Intelligence **302**, 103602 (2022)
14. Lian, J., Wang, D., Zhu, S., Wu, Y., Li, C.: Transformer-based attention network for vehicle re-identification. Electronics **11**(7), 1016 (2022)
15. Lu, Z., Lin, R., Hu, H.: Mart: Mask-aware reasoning transformer for vehicle re-identification. IEEE Transactions on Intelligent Transportation Systems (2022)
16. Ma, H., Li, X., Yuan, X., Zhao, C.: Denseformer: A dense transformer framework for person re-identification. IET Computer Vision (2022)
17. Manigrasso, F., Miro, F.D., Morra, L., Lamberti, F.: Faster-ltn: a neuro-symbolic, end-to-end object detection architecture. In: Artificial Neural Networks and Machine Learning–ICANN 2021: 30th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 14–17, 2021, Proceedings, Part II 30. pp. 40–52. Springer (2021)
18. Mao, J., Gan, C., Kohli, P., Tenenbaum, J.B., Wu, J.: The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. In: International Conference on Learning Representations. International Conference on Learning Representations, ICLR (2019)
19. Martone, S., Manigrasso, F., Lamberti, F., Morra, L.: Prototypical logic tensor networks (proto-ltn) for zero shot learning. In: 2022 26th International Conference on Pattern Recognition (ICPR). pp. 4427–4433. IEEE (2022)
20. Park, J., Bu, S.J., Cho, S.B.: A neuro-symbolic ai system for visual question answering in pedestrian video sequences. In: International Conference on Hybrid Artificial Intelligence Systems. pp. 443–454. Springer (2022)

21. Piano, L., Pratticò, F.G., Russo, A.S., Lanari, L., Morra, L., Lamberti, F.: Bent & broken bicycles: Leveraging synthetic data for damaged object re-identification. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 4881–4891 (2023)
22. Serafini, L., Donadello, I., Garcez, A.d.: Learning and reasoning in logic tensor networks: theory and application to semantic image interpretation. In: Proceedings of the Symposium on Applied Computing. pp. 125–130 (2017)
23. Silver, T., Athalye, A., Tenenbaum, J.B., Lozano-Pérez, T., Kaelbling, L.P.: Learning neuro-symbolic skills for bilevel planning. In: Conference on Robot Learning. pp. 701–714. PMLR (2023)
24. Wang, H., Shen, J., Liu, Y., Gao, Y., Gavves, E.: Nformer: Robust person re-identification with neighbor transformer. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7297–7307 (2022)
25. Wu, T., Tjandrasuwita, M., Wu, Z., Yang, X., Liu, K., Sosic, R., Leskovec, J.: Zeroc: A neuro-symbolic model for zero-shot concept recognition and acquisition at inference time. Advances in Neural Information Processing Systems **35**, 9828–9840 (2022)
26. Yu, Z., Pei, J., Zhu, M., Zhang, J., Li, J.: Multi-attribute adaptive aggregation transformer for vehicle re-identification. Information Processing & Management **59**(2), 102868 (2022)
27. Zhu, K., Guo, H., Zhang, S., Wang, Y., Liu, J., Wang, J., Tang, M.: Aaformer: Auto-aligned transformer for person re-identification. IEEE Transactions on Neural Networks and Learning Systems (2023)