Randomly Weighted, Untrained Neural Tensor Networks Achieve Greater Relational Expressiveness

Jinyung Hong 1  Theodore P. Pavlic 1

Abstract

Neural Tensor Networks (NTNs), which are structured to encode the degree of relationship among pairs of entities, are used in Logic Tensor Networks (LTNs) to facilitate Statistical Relational Learning (SRL) in first-order logic. In this paper, we propose Randomly Weighted Tensor Networks (RWTNs), which incorporate randomly drawn, untrained tensors into an NTN encoder network with a trained decoder network. We show that RWTNs meet or surpass the performance of traditionally trained LTNs for Semantic Image Interpretation (SII) tasks that have been used as a representative example of how LTNs utilize reasoning over first-order logic to exceed the performance of solely data-driven methods. We demonstrate that RWTNs outperform LTNs for the detection of the relevant part-of relations between objects, and we show that RWTNs can achieve similar performance as LTNs for object classification while using fewer parameters for learning.

1. Introduction

Combining knowledge-representation-and-reasoning techniques with artificial neural networks has the promise of enhancing the high performance of modern artificial intelligence (AI) with explainability and interpretability, which are necessary for generalized human insight and increased trustworthiness. Several recent studies across statistical relational learning (SRL), neural-symbolic computing, knowledge completion, and approximate inference (Koller et al., 2007; Garcez et al., 2008; Pearl, 2014; Nickel et al., 2015) have shown that neural networks can be integrated with logical systems to achieve robust learning and efficient inference as well as the interpretability provided by symbolic knowledge. These approaches represent knowledge in symbolic form and then use neural networks to implement logical calculus (either exactly or approximately) and thus provide a mapping between interpretable symbolism and flexible connectionism.

Several approaches to neural-network knowledge representation make use of relational embedding, which represents relational predicates in a neural network (Sutskever & Hinton, 2009; Bordes et al., 2011; Socher et al., 2013; Santoro et al., 2017). For example, Neural Tensor Networks (NTNs) are structured to encode the degree of relationship among pairs of entities in the form of tensor operations on real-valued vectors (Socher et al., 2013). These NTNs have been synthesized with neural symbolic integration (Garcez et al., 2008) in the development of Logic Tensor Networks (LTNs) (Serafini & Garcez, 2016), which are able to extend the power of NTNs to reason over first-order many-valued logic (Bergmann, 2008). Given data available in the form of real-valued vectors, LTNs allow for compactly defining logical soft and hard constraints and relationships that apply to certain subsets of the vectors in first-order logic. LTNs define basic syntax and semantics about how to map logical terms to numerical values, enabling learning effectively in hybrid domains where elements are composed of both numerical and relational information.

In this paper, we propose Randomly Weighted Tensor Networks (RWTNs), a novel NTN-based network for relational embedding that incorporates randomly drawn, untrained tensors as an NTN encoder network with a trained decoder network. Our approach is motivated by the basic architecture of an LTN combined with insights from Reservoir Computing (RC) (Jaeger, 2001), which was more traditionally applied to classification problems and time-series analysis. A conventional LTN would incorporate an NTN specially trained to capture first-order logical relationships present in data. In our case, the NTN we use is selected not by training but through the use of a 3-dimensional randomly weighted tensor acting as a generic encoder network to provide a nonlinear embedding of latent relationships among real-valued vectors. We show that a trained decoder network in RWTNs can effectively capture the likelihood of part-of relationships at a level of performance exceeding that of traditional
LTNs even if fewer parameters have to be learned. Thus, even though it is untrained, the randomly drawn NTN is shown to have great relational expressiveness and acts as a kind of general-purpose feature extractor the same way a randomly drawn recurrent reservoir in RC generates features for time-series data. To the best of our knowledge, this is the first research to integrate both RC and SRL approaches for reasoning under uncertainty and learning in the presence of data and rich knowledge.

This paper is organized as follows. We review related work in section 2 and contrast that work with the approach we take with RWTNs. We then present preliminary background information in Reservoir Computing and Logic Tensor Networks in section 3. In Section 4, we formally introduce the mathematical and structural definitions of RWTNs and discuss the theory of how these properties can increase relational expressiveness in the model. Then, in section 5, we evaluate the performance of RWTNs relative to LTNs using Semantic Image Interpretation (SII) tasks that have been used in the past to show the merits of LTNs over other purely data-driven approaches. Finally, in section 6, we give closing remarks and discuss directions for future work.

2. Related Work

As described in section 1, RWTNs are greatly influenced by LTNs and can be viewed as a performance-based refactoring of the neural network architecture. The model theory underlying LTNs was first proposed by Guha (2015); it represents logical terms and predicates using points/vectors in an n-dimensional real space and computes the truth value of atomic formulas by comparing the projections of the real-valued vector. By extending the theory and generalizing NTNs (Socher et al., 2013), LTNs (Serafini & Garcez, 2016) (and thus also RWTNs) provide more general interpretation of predicate symbols in first-order logic. Because of the strong relationship between LTNs and RWTNs, we compare performances between LTNs and RWTNs, we compare performances between LTNs and RWTNs directly in section 5, and we show that RWTNs meet or surpass the performance of LTNs for certain tasks although having fewer parameters to learn.

Another neural-network approach for logical representation comes from (Hybrid) Markov Logic Networks (MLNs) (Richardson & Domingos, 2006; Wang & Domingos, 2008; Nath & Domingos, 2015). In MLNs, the number of models that satisfy a formula determines the truth value of the formula. That is, the more models there are, the higher the degree of truth. Hybrid MLNs introduce a dependence from real features associated to constants, which is given and not learned. In our model, instead, the truth value of a complex formula is determined by (fuzzy) logical reasoning, and the relations between the features of different objects is learned through error minimization.

A tensor factorization and Bayesian clustering approach for relational learning is presented by Sutskever & Hinton (2009). They cluster the entities in a nonparametric Bayesian framework, whereas our model relies solely on learned entity vectors. Furthermore, they use MCMC for inference and learning while our model uses standard forward propagation and backpropagation techniques.

Our model can be related to other models in different fields. In deep learning literature, a factored 3-way Restricted Boltzmann Machine parameterized by a tensor is proposed by Ranzato et al. (2010). Moreover, Yu et al. (2012) introduced a tensor layer model for speech recognition. However, their models are only applicable inside deeper neural networks and are a special case of NTNs.

3. Preliminaries

3.1. Reservoir Computing

Reservoir Computing (RC) is a less conventional method for using Recurrent Neural Networks that has been widely used in applications such as time-series forecasting (Deihimi & Showkati, 2012; Bianchi et al., 2015b;a), process modelling (Rodan et al., 2017), speech analysis (Trentin et al., 2015), and classification of multivariant time series (Bianchi et al., 2018). RC models conceptually divide time-series processing into two components: (i) representation of temporal structure in the input stream through a non-adaptable dynamic reservoir (generated through the feedback-driven dynamics of a randomly drawn RNN), and (ii) an easy-to-adapt readout from the reservoir. The feedbacks within the reservoir network provide internal dynamic state variables allowing the network to re-shape and extend the duration of short patterns in time, effectively allowing the readout network to have access to “echoes” of past versions of the input data. Consequently, RC techniques were originally introduced to the machine learning community under the name Echo State Networks (ESNs) (Jaeger, 2001); in this paper, we use the two terms interchangeably.

Similar to conventional methods for constructing a sequence model in machine learning, an ESN can be defined as the combination of an encoding and a decoding function. The encoder is used to produce a representation of the input, whereas the decoder is a discriminative (or predictive) model that calculates the posterior probability of the output given the representation provided by the encoder. The simplest formulation of the recurrent mapping from input to the internal state of the ESN is:

$$h(t) = f(W_{in}x(t) + W_{h}h(t-1))$$  \hspace{1cm} (1)$$

where $h(t)$ is the internal state of the ESN at time $t$, which depends upon its previous state $h(t-1)$ and the current input $x(t)$ by way of $f(\cdot)$, a nonlinear activation function (usually
a sigmoid or hyperbolic tangent), and the encoder parameters \( \{W_{in}, W_r\} \) that are randomly generated and left untrained (or implemented using a prefixed topology (Rodan & Tino, 2010)).

Although the mapping from input to the internal state of the ESN is untrained, the rich dynamics of the large recurrent layer (reservoir) have the potential to extract useful features from time-series data. The favorable capabilities of the reservoir primarily depend on three factors: (i) a large number of processing units in the recurrent layer, (ii) random connectivity of the recurrent layer, and (iii) a spectral radius\(^1\) of the connection weights matrix \( W_r \), set to bring the system to the edge of stability (Bianchi et al., 2016). Therefore, rather than training the internal weight matrices, the behavior of the reservoir can be controlled by simply modifying: the spectral radius \( \rho \), the percentage of non-zero connections \( \beta \), and the number of hidden units \( R \). Another important hyperparameter is the scaling \( \omega \) of the values in \( W_{in} \), which controls the degree of nonlinearity in the processing units and, jointly with \( \rho \), can change the internal dynamics from a chaotic system to a contractive one. Finally, for the purpose of regularization, a Gaussian noise with standard deviation \( \xi \) can be added to the state update function (Eq.(1)) as an argument (Jaeger, 2001).

From the sequence of the ESN states generated over time, described by the matrix \( H = [h(1),...,h(T)]^T \), it is possible to define an encoding (representation) \( r(H) = r_x \) of the input sequence \( x \). Such a state becomes a vector representation with a fixed-size and can be processed by regular machine learning algorithms. Specifically, the decoder maps the input representation \( r_x \) into the output space containing all class labels \( y \) in a classification task:

\[
y = g(r_x) = V_o r_x + v_o \tag{2}
\]

The decoder parameters \( \{V_o, v_o\} \) can be learned by minimizing a ridge regression loss function

\[
\{V_o, v_o\}^* = \arg \min_{\{V_o, v_o\}} \frac{1}{2} ||V_o r_x + v_o - y||^2 + \lambda ||V_o||^2 \tag{3}
\]

which admits a closed-form solution (Scardapane & Wang, 2017).

However, a simple linear model may not possess sufficient representational capacity to model the high-level embeddings derived from the reservoir states. For this reason, several authors have proposed replacing the standard linear decoding function \( g(\cdot) \) in Eq.(2) with a nonlinear model, such as support vector machines (SVMs) (Li et al., 2012; Bianchi et al., 2015b) or multi-layer perceptrons (MLPs) (Maass et al., 2002; Bush & Anderson, 2005; Babinec & Pospíchal, 2006). In particular, MLP is an universal function approximator that can learn complex input representations by stacking multiple layers of neurons configured with nonlinear activation functions, e.g., rectified linear units (ReLUs). Deep MLPs are known for their ability to dissemble factors of variations from high-dimensional feature spaces (Goodfellow et al., 2009) and therefore can be more powerful and articulate through their mappings from the representation to the output space than linear readouts.

As we describe in section 4, we effectively make use of RC without the recurrence that would otherwise be a key defining feature of the dynamic reservoirs. We use guidelines from RC for constructing the randomly drawn input matrix \( W_{in} \), while we omit an implementation of the feedback matrix \( W_r \), thereby removing the internal state from our degenerate reservoirs. In the future, when we intend to develop systems for reasoning over time-series data (e.g., for evaluating temporal logic relationships), we will investigate re-introducing recurrence. For our current focus on relationships defined at only a single time, we only inherit the randomly weighted input matrix from RC.

### 3.2. Logic Tensor Networks

Logic Tensor Networks integrate learning based on NTNs (Socher et al., 2013) with reasoning using first-order, many-valued logic (Bergmann, 2008), all implemented in TENSORFLOW\textsuperscript{TM} (Serafini & Garcez, 2016). This enables a range of knowledge-based tasks using rich knowledge representation in First-Order Logic (FOL) to be combined with efficient data-driven machine learning. Following the presentation of Serafini & Garcez (2016), this section briefly introduces the syntax and semantics in LTNs for the use of mapping logical symbols to numerical values and learning reasoning relations among real-valued vectors using the logical formulas.

**First-Order Logic.** A FOL language \( \mathcal{L} \) and its signature consists of three disjoint sets—\( \mathcal{C}, \mathcal{F} \) and \( \mathcal{P} \)—denoting constants, functions and predicate symbols, respectively. For any function or predicate symbol \( s, \alpha(s) \) can be described as its *arity*. Logical formulas in \( \mathcal{L} \) enable the description of relational knowledge. For example, if the atomic formula \( \text{partOf}(o_1, o_2) \) denotes that object \( o_1 \) is a part of object \( o_2 \), then the quantification \( \forall x (\text{Cat}(x) \rightarrow \exists y (\text{partOf}(x, y) \wedge \text{Tail}(y))) \) denotes that every cat should have a tail.

The objects being reasoned over with FOL are mapped to an interpretation domain, which is a subset of \( \mathbb{R}^n \) so that every object is associated with an \( n \)-dimensional vector of real numbers. Intuitively, this \( n \)-tuple indicates \( n \) numerical features of an object. Thus, functions are interpreted as real-valued functions, and predicates are interpreted as fuzzy relations on real vectors. With this numerical background,
we can now define the numerical grounding of FOL with the following semantics; this grounding is necessary for NTNs to reason over logical statements.

**Definition 1.** Let \( n \in \mathbb{N} \). An \( n \)-grounding, \( \hat{G} \) for a FOL \( \mathcal{L} \) is a function defined on the signature of \( \mathcal{L} \) satisfying the following conditions:

1. \( \hat{G}(c) \in \mathbb{R}^n \) for every constant symbol \( c \in \mathcal{C} \);
2. \( \hat{G}(f) \in \mathbb{R}^{n \cdot \alpha(f)} \) for function symbol \( f \in \mathcal{F} \);
3. \( \hat{G}(P) \in \mathbb{R}^{n \cdot \alpha(P)} \) for predicate sym. \( P \in \mathcal{P} \);

Given a grounding \( \hat{G} \), the semantics of closed terms and atomic formulas is defined as follows:

\[
\hat{G}(f(t_1, \ldots, t_m)) = \hat{G}(f)(\hat{G}(t_1), \ldots, \hat{G}(t_m))
\]

\[
\hat{G}(P(t_1, \ldots, t_m)) = \hat{G}(P)(\hat{G}(t_1), \ldots, \hat{G}(t_m))
\]

According to fuzzy logic such as the Lukasiewicz \( t \)-norm (Bergmann, 2008), the semantics for connectives is defined as follows:

\[
\hat{G}(\neg \phi) = 1 - \hat{G}(\phi)
\]

\[
\hat{G}(\phi \land \psi) = \max(0, \hat{G}(\phi) + \hat{G}(\psi) - 1)
\]

\[
\hat{G}(\phi \lor \psi) = \min(1, \hat{G}(\phi) + \hat{G}(\psi))
\]

\[
\hat{G}(\phi \rightarrow \psi) = \min(1, 1 - \hat{G}(\phi) + \hat{G}(\psi))
\]

**Learning as Best Satisfiability.** A partial grounding \( \hat{G} \) can be defined on a subset of the signature of \( \mathcal{L} \). A grounding \( \hat{G} \) is said to be a completion of \( \hat{G} \) if \( \hat{G} \) is a grounding for \( \mathcal{L} \) and coincides with \( \hat{G} \) on the symbols where \( \hat{G} \) is defined.

**Definition 2.** Let \( \langle K, \hat{G} \rangle \) be a grounded theory which is a pair \( \langle K, \hat{G} \rangle \) with a set \( K \) of closed formulas and a partial grounding \( \hat{G} \).

**Definition 3.** A grounding \( \hat{G} \) satisfies a \( \langle K, \hat{G} \rangle \) if \( \hat{G} \) completes \( \hat{G} \) and \( \hat{G}(\phi) = 1 \) for all \( \phi \in K \). A \( \langle K, \hat{G} \rangle \) is satisfiable if there exists a grounding \( \hat{G} \) that satisfies \( \langle K, \hat{G} \rangle \).

According to the previous definition, deciding the satisfiability of \( \langle K, \hat{G} \rangle \) amounts to searching for a grounding \( \hat{G} \) such that all the formulas of \( K \) are mapped to 1. Differently from classical satisfiability, when a \( \langle K, \hat{G} \rangle \) is not satisfiable, we are interested in the best possible satisfaction that we can reach with a grounding. This is defined as follows.

**Definition 4.** Let \( \langle K, \hat{G} \rangle \) be a grounded theory. We define the best satisfiability problem as the problem of finding a grounding \( \hat{G}^* \) that maximizes the truth values of the conjunction of all clauses \( cl \in K \), i.e.,

\[
\hat{G}^* = \arg \max_{\hat{G} \in \mathcal{G}} \hat{G}(\bigwedge_{cl \in K} cl).
\]

**Logical Grounding and NTNs.** Grounding \( \hat{G}^* \) captures the implicit correlation between quantitative features of objects and their categorical/relational properties. We consider groundings of the following form.

Function symbols are grounded to linear transformations. If \( f \) is a \( m \)-ary function symbol, then \( \hat{G}(f) \) is of the form:

\[
\hat{G}(f)(v) = M_f v + N_f
\]

where \( v = (v_1^T, \ldots, v_m^T)^T \) is the \( mn \)-ary vector obtained by concatenating each \( v_i \). The parameters for \( \hat{G}(f) \) are the \( n \times mn \) real matrix \( M_f \) and the \( n \)-vector \( N_f \).

The grounding of an \( m \)-ary predicate \( P \), namely \( \hat{G}(P) \), is defined as a generalization of the NTN (Socher et al., 2013), as a function from \( \mathbb{R}^{mn} \) to \( [0, 1] \), as follows:

\[
\hat{G}(P)(v) = \sigma(u^T P(v) v + V_P v + b_P))
\]

where \( \sigma \) is the sigmoid function and \( P \) is the hyperbolic tangent (\( \tanh \)). The parameters for \( P \) are: \( W_P^{[1:k]} \), a 3-D tensor in \( \mathbb{R}^{k \times mn \times mn} \), \( V_P \in \mathbb{R}^{k \times mn} \), \( b_P \in \mathbb{R}^{k} \) and \( u_P \in \mathbb{R}^{k} \). This last parameter computes a linear combination of the quadratic features given by the tensor product. With this encoding, the grounding (i.e., truth-value) of a clause can be calculated by a neural network which first computes the grounding of the literals contained in the clause, and then combines them using the specific \( t \)-norm.

**4. Randomly Weighted Tensor Networks.**

In this section, we introduce the mathematical and structural definitions of Randomly Weighted Tensor Networks (RWTNs). By combining a randomly drawn, untrained tensor into an NTN encoder network with a trained decoder network, our model not only has fewer parameters to learn, but also can achieve greater expressible capability for extracting relational knowledge as an LTN trained for the same task.

RWTNs can be defined as a function from \( \mathbb{R}^{mn} \) to \( [0, 1] \):

\[
\hat{G}_{rwt}(P)(v) = \sigma(k^T \xi (v^T W^{[1:k]}(v) v + V_{in} v + \xi))
\]

where \( \sigma \) is the sigmoid function and \( \xi \) is the hyperbolic tangent (\( \tanh \)) function. The parameters of the RWTN encoder include: \( W^{[1:k]} \in \mathbb{R}^{mn \times mn \times R} \) (a 3-dimensional randomly weighted tensor), \( V_{in} \in \mathbb{R}^{R \times mn} \) (randomly drawn input-layer weights), and \( \xi \) (Gaussian noise). The parameters of the RWTN decoder are thus \( u \in \mathbb{R}^{R \times d} \) and \( k \in \mathbb{R}^d \), which are the standard weights for a single hidden layer neural network where \( t \) is the number of neurons in a hidden layer.

Fig.1 shows a sample visualization of the structure of our model. In the depicted case, \( e_1, e_2 \in \mathbb{R}^d \) are vector representations (or features) of two entities for which the RWTN expresses some level relationship between. Each slice of the tensor \( W^{[1:k]} \) can be viewed as being responsible for representing one kind of relationship between the two entities. In principle, the network could.
be trained to explicitly represent certain relationships, such as \((e_1, R, e_2) = (\text{cat}, \text{has part}, \text{tail})\). However, this tensor is randomly weighted in RWTN to span a wide range of potential relationships that are left to the later decoder to mix to represent the desired relationships from data.

The first underlying characteristic of our model is the random and non-adaptable property of the parameters \(\{W_{res}^{[1:R]}, V_{in}, \xi\}\) in the encoder network, inspired by the insights of Reservoir Computing (RC). Strictly speaking, the weights of \(W_{res}^{[1:R]}\) are not arbitrarily drawn. The weights are drawn according to hyperparameters defined by RC; in particular, they are selected to have: a greater number of units, random sparsity, and a certain spectral radius. We intend that by having those properties, the randomly weighted tensor in our model can act as a filter that converts the latent relationship between objects using a higher-dimensional map, similar to the operation of an explicit, temporal kernel function.

The insight of the internal weight in RC is also applied similar to the operation of an explicit, temporal kernel function, to the input layer parameter \(V_{in}\) in our model. The input weights are (as usual) generated randomly from a uniform distribution over an interval \([-\omega, \omega]\). Furthermore, the sign of each input weight is determined by a random draw from Bernoulli distribution (input-layer weights in Fig.1). These characteristics are also expected to play an important role in having more expressiveness for extracting relational knowledge. A Gaussian noise \(\xi\) is used for the same purpose of the one in RC, which is regularization.

Succinctness in learning process of a decoder network is the second characteristic of our model. Using a single hidden layer neural network as a decoder enables learning the mapping mechanism between the high-level relational representation from the encoder and the degree of relationship among input even though fewer parameters are employed for learning \((k, m)\) compared to the conventional neural tensor networks (hidden and output layers in Fig.1).

## 5. Experimental Evaluation

To evaluate the performance of our proposed RWTNs over LTNs, we employ both for Semantic Image Interpretation (SII) tasks, which extract structured semantic descriptions from images. Very few SRL applications have been applied to SII tasks because of the high complexity involved with image learning. Donadello et al. (2017) define two main tasks of SII as: (i) the classification of bounding boxes, and (ii) the detection of the part-of relation between any two bounding boxes. They demonstrated that LTNs can successfully improve the performance of solely data-driven approaches, including the state-of-the-art Fast Region-based Convolutional Neural Networks (Fast R-CNN) (Girshick, 2015). Moreover, they showed that LTNs effectively handle noisy data through the systematic creation of training sets with errors in the labels. Our experiments are conducted by comparing the performance of two tasks of SII between RWTNs and the tensor networks in LTNs from Donadello et al. (2017); these tasks are well defined in first-order logic, and the codes implemented in TENSORFLOW™ have been provided and thus can be easily used to compare the performance of LTNs with our proposed RWTNs.

### 5.1. Methods

Here, we provide details of our experimental comparison of RTWNs and LTNs. Sections 5.1.1 and 5.1.2 introduce how to formalize our two focal SII tasks in FOL grounded in RWTNs and LTNs in a manner similar to that of Donadello et al. (2017). Next, in section 5.1.3, we describe the data set used in the test. Then, in section 5.1.4, we describe the RWTN and LTN hyperparameters used in the test.

#### 5.1.1. Formalizing SII in First-Order Logic

A signature \(\Sigma_{SII} = \langle C, F, P \rangle\) is defined where \(C = \bigcup_{p \in P_{unary}} b(p)\) is the set of identifiers for all the bounding boxes in all the images, \(F = \emptyset\), and \(P = \{P_1, P_2\}\), where \(P_1\) is a set of unary predicates, one for each object type (e.g., \(P_1 = \{\text{Dog}, \text{Cat}, \text{Tail}, \ldots\}\)), and \(P_2\) is a set of binary predicates representing relations between objects. Because our experiments focus on the part-of relation, \(P_2 = \{\text{partOf}\}\). The FOL formulas based on this signature can specify: (i) simple facts (e.g., the fact that bounding box \(b\) contains a cat, written \(\text{Cat}(b)\); the fact that \(b\) contains either a cat or a dog, written \(\text{Cat}(b) \lor \text{Dog}(b)\); etc.), or (ii) general rules (e.g., \(\forall x(\text{Cat}(x) \rightarrow \exists y(\text{partOf}(x, y) \land \text{Tail}(y)))\)).
We define the grounding for $\Sigma_{SII}$ such that each constant $b$, indicating a bounding box, is associated with geometric features describing the position and the dimension of the bounding box and semantic features indicating the classification score returned by the bounding box detector for each class. For example, for each bounding box $b \in \mathcal{C}$, $G(b)$ is the vector:

$$G(b) = (1, \ldots, x_0(b), y_0(b), x_1(b), y_1(b))$$

where the last four features are the coordinates of the top-left and bottom-right corners of $b$, and $class(C_i, b) \in [0, 1]$ is the classification score of the bounding box detector for $b$.

An example of groundings for predicates can be defined by taking a one-vs-all multi-classifier approach as follows. First, for each class $C_i \in \mathcal{P}$, define the grounding:

$$G(C_i)(x) = \begin{cases} 1, & \text{if } i = \arg \max_{1 \leq l \leq |\mathcal{P}|} x_l \\ 0, & \text{otherwise} \end{cases}$$

where $x = \langle x_1, \ldots, x_{4+|\mathcal{P}|} \rangle$ is the vector corresponding to the grounding of a bounding box. Then, a simple rule-based approach for defining a grounding for the $\text{partOf}$ relation is based on the naive assumption that the more a bounding box $b$ is contained within a bounding box $b'$, the higher the probability should be that $b$ is part of $b'$. Accordingly, one can define $G(\text{partOf}(b, b'))$ as the inclusion ratio $ir(b, b')$ of bounding box $b$, with grounding $x$, into bounding box $b'$, with grounding $x'$ (formally, $ir(b, b') = \frac{\text{area}(b)}{\text{area}(b')} \cdot \frac{\text{area}(b)}{\text{area}(b')}$). A slightly more sophisticated rule-based grounding for $\text{partOf}$ (used as baseline in the experiments to follow) takes into account also type compatibilities by multiplying the inclusion ratio by a factor $w_{ij}$. Hence, $G(\text{partOf}(b, b'))$ can be defined as:

$$G(\text{partOf}(b, b')) = \begin{cases} 1, & \text{if } ir(b, b') \cdot \max_{1 \leq l \leq |\mathcal{P}|} (w_{ij} \cdot x_i \cdot x'_j) \geq th_{ir} \\ 0, & \text{otherwise} \end{cases}$$

for some threshold $th_{ir}$ (usually, $th_{ir} > 0.5$) where $w_{ij} = 1$ if $C_i$ is a part of $C_j$ ($w_{ij} = 0$ otherwise). Given the above grounding, we can compute the grounding of any atomic formula thus expressing the degree of truth of the formula.

5.1.2. DEFINING THE GROUNDED THEORIES FOR RWTNs AND LTNs

A suitable ground theory $GT$ can be built for SII. Let $\mathcal{P}ics^I \subseteq \mathcal{P}ics$ be a set of bounding boxes of images correctly labelled with the classes that they belong to, and let each pair of bounding boxes be correctly labelled with the $\text{partOf}$ relation. Then, $\mathcal{P}ics^I$ can be considered as a training set and a grounded theory $\mathcal{T}_{\text{RWTN}}$ can be constructed as follows: $\mathcal{T}_{\text{RWTN}} = \langle \mathcal{K}, \hat{G} \rangle$, where:

- $\mathcal{K}$ contains the two following sets: (i) the set of closed literals $C_i(b)$ (resp. $\neg C_i(b)$) and $\text{partOf}(b, b')$ (resp. $\neg \text{partOf}(b, b')$) for every bounding box $b$ labelled (resp. not labelled) with $C_i$ and for every pair of bounding boxes $\{b, b'\}$ connected (resp. not connected) by the $\text{partOf}$ relation, and (ii) the set of the mereological constraints for the $\text{partOf}$ relation, including asymmetric constraints ($\forall xy(\text{partOf}(x, y) \rightarrow \neg \text{partOf}(y, x))$), lists of several parts of an object (e.g., $\forall xy(\text{Cat}(x) \land \text{partOf}(x, y) \rightarrow \text{Tail}(y) \lor \text{Muzzle}(y))$), or restrictions that whole objects cannot be part of other objects (e.g., $\forall xy(\text{Cat}(x) \rightarrow \neg \text{partOf}(y, x))$) and every part object cannot be divided further into parts (e.g., $\forall xy(\text{Tail}(x) \rightarrow \neg \text{partOf}(y, x))$).

- The partial grounding $\hat{G}$ is defined on all bounding boxes of all the images in $\mathcal{P}ics$ where both $\text{class}(C_i, b)$ and the bounding box coordinates are computed by the Fast R-CNN object detector. $\hat{G}$ is not defined for the predicate symbols in $\mathcal{P}$ and is to be learned.

A grounded theory $\mathcal{T}_{\text{LTN}}$ is only slightly different. $\mathcal{T}_{\text{LTN}} = \langle \mathcal{K}, \hat{G}_{\text{rwtn}} \rangle$ where a partial grounding $\hat{G}_{\text{rwtn}}$ can be described for predicates using eq.(5). Thus, we can easily compare the performance between $\hat{G}_{\text{rwtn}}$ and $\hat{G}$.

5.1.3. DATASETS

The PASCAL-PART-dataset (Chen et al., 2014) and ontologies (WORDNET) are chosen for the $\text{partOf}$ relation. The PASCAL-PART-dataset contains 10103 images with bounding boxes. They are annotated with object-types and the part-of relation defined between pairs of bounding boxes. There are three main groups in labels—animals, vehicles, and indoor-objects—with their corresponding parts and "part-of" label. There are total 59 labels (20 labels for whole objects and 39 labels for parts).

For the purpose of data pre-processing, data samples with bounding boxes with height or width smaller than 6 pixels have been omitted. The images were then split into a training set with 80% of the images and a test set with 20% of the images, maintaining the same proportion of the number of bounding boxes for each label. Given a set of bounding boxes detected by an object detector (Fast-RCNN), the task of object classification is to assign to each bounding box an object type. The task of $\text{partOf}$ detection is to decide, given two bounding boxes, if the object contained in the first is a part of the object contained in the second.

5.1.4. HYPERPARAMETER SETTING

To compare the performance between RWTNs and LTNs, we train two models separately.
• For RWTN, the random coarse search is used for finding the hyperparameters. The spectral radius \( \rho \) is set to 0.6, the connection sparsity \( \beta \) is 0.25. The size of the reservoir \( R \) is 200. The input scaling \( \omega \) is 0.5. The noise level \( \xi \) is 0.01. The number of hidden units for a readout \( t \) is 20.

• For LTN, we configure the experimental environment following Donadello et al. (2017). The LTNs were configured with a tensor of \( k = 6 \) layers.

Based on Donadello et al. (2017), both models make use of a regularization parameter \( \lambda = 10^{-10} \), Lukasiewicz \( t \)-norm \((\mu(a, b) = \max(0, a + b - 1))\), and the harmonic mean as an aggregation operator. We ran 1000 training epochs of the RMSProp learning algorithm available in TENSORFLOW\textsuperscript{TM} for each model.

5.2. Results

Our experiments mainly focus on the comparison of the performance between our model and LTN, but figures also include the results with the Fast-RCNN (Girshick, 2015) at type classification (Eq.(6)) and the inclusion ratio \( ir \) baseline (Eq.(7)) at the part-of detection task. If \( ir \) is greater than a given threshold \( th \) (in our experiments, \( th = 0.7 \)), then the bounding boxes are said to be in the part-of relation. Every bounding box \( b \) is classified into \( C \in P_1 \) if \( G(C(b)) > th \).

Results for indoor objects are shown in Fig.2 where AUC is the area under the precision–recall curve. The results show that, for the part-of relation, RWTNs achieve better performance than LTNs. Even though RWTNs has slightly worse performance than LTNs for object types, RWTNs also improve the performance of the Fast-RCNN (FRCNN) object detector. In the Fig.3, RWTNs perform better than LTNs for the detection of the part-Of relation on vehicle and animal objects, respectively.

Although the above figures show the good comparison of the performances of the models, there are some variance in the results because of the stochastic nature of the experiments. Consequently, we carried out five such experiments for each task, for which the sample averages and 95% confidence intervals are shown in Table 1. These results confirm that the outcomes shown in Figures 2 and 3 are typical, and our model can achieve similar performance as LTNs for object-task classification and superior performance for detection of part-of relations.

5.3. Relative Complexity of RWTNs and LTNs

To better appreciate the relative performance of RWTNs and LTNs, we can compare the number of parameters to learn for grounding a unary predicate for each model. Let \( n \) be the number of features of an input \((n = 65)\). As shown in Eq.(4), the parameters to learn in LTNs are \( \{u_P \in \mathbb{R}^k, W_P^{[1:k]} \in \mathbb{R}^{n \times n \times k}, V_P \in \mathbb{R}^{k \times n}, b_P \in \mathbb{R}^k\} \), where \( k = 6 \) following the configuration of the LTNs. Thus, the number of parameters in LTNs is \( k + n \cdot k + k \cdot n + k = (n^2 + n + 2) \cdot k = (65^2 + 65 + 2) \cdot 6 = 25752 \).

On the other hand, in Eq.(5), the learnable parameters in RWTNs are only \( \{k \in \mathbb{R}, u \in \mathbb{R}^{R \times 1}\} \), where \( R = 200 \) and \( t = 20 \) following the configuration of the RWTNs. Therefore, the number of learnable parameters in RWTNs is \( t + R \cdot t = (R + 1) \cdot t = 201 \cdot 20 = 4020 \). The fact that the number of parameters to learn in RWTNs (4020) is significantly smaller compared to LTNs (25752) shows that non-adaptable parameters in RWTNs can have significant power to represent the latent relationship among objects so that the model can efficiently extract relational knowledge even though using fewer parameters. In other words, it may be very likely to randomly draw an NTN encoder in RWTNs whose expressiveness rivals or greatly exceeds any locally optimal NTN encoder discovered through backpropagation with a traditional LTN.

Furthermore, the number of the parameters of LTNs heavily depends on the number of features, whereas RWTNs are independent of the number of features. In principle, this could allow the learning process in our model to be accelerated if the feature representation from the encoder model is pre-processed and stored. A more sophisticated analysis related to the correlation between the complexity of learning and the layer size remains for our future work.

6. Conclusion and Future Work

In this paper, we introduced Randomly Weighted Tensor Networks, which, when compared to a conventional neural tensor model, act as a generalized feature extractor with greater relational expressiveness and a learning model with relatively simpler structure. We demonstrated how insights from Reservoir Computing normally reserved for time-series analysis can be applied to the fields of neural-symbolic computing and knowledge representation and reasoning for relational learning.
Randomly Weighted, Untrained Neural Tensor Networks Achieve Greater Relational Expressiveness

(a) LTNs slightly outperform RWTNs on object type classification, achieving an Area Under the Curve (AUC) of 0.782 in comparison with 0.777. RWTNs however improve the performance of Fast R-CNN.

(b) RWTNs show better performance than LTNs in the detection of part-of relations, achieving AUC of 0.661 in comparison with 0.620.

Figure 2. Precision–recall curves for indoor objects type classification and the partOf relation between objects.

(a) RWTNs outperform LTNs for the detection of part-of relations among vehicle objects, achieving AUC of 0.581 compared to 0.471.

(b) RWTNs perform better than LTNs in the task of part-of relations detection among animal objects, achieving AUC of 0.667 in comparison with 0.634.

Figure 3. Precision–recall curves for the partOf relation between vehicle and animal objects, respectively.

Our work can be advanced in several ways. We will develop a novel application of RWTNs for challenging problems such as visual question-answering tasks that need to extract structural knowledge from not only images but also text. In addition, we will investigate how other methods from reservoir computing for exploring efficient reservoir topologies (van der Zant et al., 2004; Schmidhuber et al., 2007; Ferreira & Ludermir, 2009; Sun et al., 2017; Wang et al., 2019) might be generalized to these new application spaces. As our model was inspired by Echo State Networks, it may be improved with theoretical foundations such as reservoir adaptation that generate more efficient reservoirs or fine-tune reservoir weights using an unsupervised learning rule for certain tasks. Finally, we shall extend RWTNs to include a recurrent part for representing dynamic features of time-series data; this approach may allow for extracting time-varying relational knowledge necessary for developing a framework for data-driven reasoning over temporal logic.
Randomly Weighted, Untrained Neural Tensor Networks Achieve Greater Relational Expressiveness

References

Babinec, Š. and Pospíchal, J. Merging echo state and feed-forward neural networks for time series forecasting. In *International Conference on Artificial Neural Networks*, pp. 367–375. Springer, 2006.

Bergmann, M. *An introduction to many-valued and fuzzy logic: semantics, algebras, and derivation systems*. Cambridge University Press, 2008.

Bianchi, F. M., De Santis, E., Rizzi, A., and Sadeghian, A. Short-term electric load forecasting using echo state networks and pca decomposition. *IEEE Access*, 3:1931–1943, 2015a.

Bianchi, F. M., Scardapane, S., Uncini, A., Rizzi, A., and Sadeghian, A. Prediction of telephone calls load using echo state network with exogenous variables. *Neural Networks*, 71:204–213, 2015b.

Bianchi, F. M., Livi, L., and Alippi, C. Investigating echo-state networks dynamics by means of recurrence analysis. *IEEE transactions on neural networks and learning systems*, 29(2):427–439, 2016.

Bianchi, F. M., Scardapane, S., Løkse, S., and Jenssen, R. Reservoir computing approaches for representation and classification of multivariate time series. *arXiv preprint arXiv:1803.07870*, 2018.

Bordes, A., Weston, J., Collobert, R., and Bengio, Y. Learning structured embeddings of knowledge bases. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

Bush, K. and Anderson, C. Modeling reward functions for incomplete state representations via echo state networks. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks*, 2005., volume 5, pp. 2995–3000. IEEE, 2005.

Chen, X., Mottaghi, R., Liu, X., Fidler, S., Urtasun, R., and Yuille, A. Detect what you can: Detecting and representing objects using holistic models and body parts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1971–1978, 2014.

Deihimi, A. and Showkati, H. Application of echo state networks in short-term electric load forecasting. *Energy*, 39(1):327–340, 2012.

Donadello, I., Serafini, L., and Garcez, A. D. Logic tensor networks for semantic image interpretation. *arXiv preprint arXiv:1705.08968*, 2017.

Ferreira, A. A. and Ludermir, T. B. Genetic algorithm for reservoir computing optimization. In *2009 International Joint Conference on Neural Networks*, pp. 811–815. IEEE, 2009.

Garcez, A. S., Lamb, L. C., and Gabbay, D. M. *Neural-symbolic cognitive reasoning*. Springer Science & Business Media, 2008.

Girshick, R. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015.

Goodfellow, I., Lee, H., Le, Q. V., Saxe, A., and Ng, A. Y. Measuring invariances in deep networks. In *Advances in neural information processing systems*, pp. 646–654, 2009.

Guha, R. Towards a model theory for distributed representations. In *2015 AAAI Spring Symposium Series*, 2015.

Jaeger, H. The echo state approach to analysing and training recurrent neural networks—with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 148(34):13, 2001.

Koller, D., Friedman, N., Džeroski, S., Sutton, C., McCallum, A., Pfeffer, A., Abbeel, P., Wong, M.-F., Heckerman, D., Meek, C., et al. *Introduction to statistical relational learning*. MIT press, 2007.

Li, D., Han, M., and Wang, J. Chaotic time series prediction based on a novel robust echo state network. *IEEE Transactions on Neural Networks and Learning Systems*, 23(5):787–799, 2012.

Maass, W., Natschläger, T., and Markram, H. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural computation*, 14(11):2531–2560, 2002.

Nath, A. and Domingos, P. M. Learning relational sum-product networks. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

Nickel, M., Murphy, K., Tresp, V., and Gabrilovich, E. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, 2015.

Pearl, J. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Elsevier, 2014.

Ranzato, M., Krizhevsky, A., and Hinton, G. Factored 3-way restricted boltzmann machines for modeling natural images. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 621–628, 2010.

Richardson, M. and Domingos, P. Markov logic networks. *Machine learning*, 62(1-2):107–136, 2006.
Randomly Weighted, Untrained Neural Tensor Networks Achieve Greater Relational Expressiveness

Rodan, A. and Tino, P. Minimum complexity echo state network. *IEEE transactions on neural networks*, 22(1): 131–144, 2010.

Rodan, A., Sheta, A. F., and Faris, H. Bidirectional reservoir networks trained using SVM+privileged information for manufacturing process modeling. *Soft Computing*, 21 (22):6811–6824, 2017.

Santoro, A., Raposo, D., Barrett, D. G., Malinowski, M., Pascanu, R., Battaglia, P., and Lillicrap, T. A simple neural network module for relational reasoning. In *Advances in neural information processing systems*, pp. 4967–4976, 2017.

Scardapane, S. and Wang, D. Randomness in neural networks: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 7(2):e1200, 2017.

Schmidhuber, J., Wierstra, D., Gagliolo, M., and Gomez, F. Training recurrent networks by evolino. *Neural computation*, 19(3):757–779, 2007.

Serafini, L. and Garcez, A. d. Logic tensor networks: Deep learning and logical reasoning from data and knowledge. *arXiv preprint arXiv:1606.04422*, 2016.

Socher, R., Chen, D., Manning, C. D., and Ng, A. Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems*, pp. 926–934, 2013.

Sun, X., Li, T., Li, Q., Huang, Y., and Li, Y. Deep belief echo-state network and its application to time series prediction. *Knowledge-Based Systems*, 130:17–29, 2017.

Sutskever, I. and Hinton, G. E. Using matrices to model symbolic relationship. In *Advances in neural information processing systems*, pp. 1593–1600, 2009.

Trentin, E., Scherer, S., and Schwenker, F. Emotion recognition from speech signals via a probabilistic echo-state network. *Pattern Recognition Letters*, 66:4–12, 2015.

van der Zant, T., Bečanović, V., Ishii, K., Kobialka, H.-U., and Ploeger, P. Finding good echo state networks to control an underwater robot using evolutionary computations. *IFAC Proceedings Volumes*, 37(8):215–220, 2004.

Wang, J. and Domingos, P. M. Hybrid markov logic networks. In *AAAI*, volume 8, pp. 1106–1111, 2008.

Wang, X., Jin, Y., and Hao, K. Echo state networks regulated by local intrinsic plasticity rules for regression. *Neurocomputing*, 351:111–122, 2019.

Yu, D., Deng, L., and Seide, F. Large vocabulary speech recognition using deep tensor neural networks. In *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.