An Insect-Inspired Randomly, Weighted Neural Network with Random Fourier Features For Neuro-Symbolic Relational Learning

Jinyung Hong[1], Theodore P. Pavlic[1,2,3,4]

[1]*School of Computing and Augmented Intelligence, Arizona State University, Tempe, AZ 85281, USA*
[2]*School of Sustainability, Arizona State University, Tempe, AZ 85281, USA*
[3]*School of Complex Adaptive Systems, Arizona State University, Tempe, AZ 85281, USA*
[4]*School of Life Sciences, Arizona State University, Tempe, AZ 85281, USA*

Abstract

The computer-science field of Knowledge Representation and Reasoning (KRR) aims to understand, reason, and interpret knowledge as efficiently as human beings do. Because many logical formalisms and reasoning methods in the area have shown the capability of higher-order learning, such as abstract concept learning, integrating artificial neural networks (ANNs) with KRR methods for learning complex and practical tasks has received much attention. For example, Neural Tensor Networks (NTNs) are neural-network models capable of transforming symbolic representations into vector spaces where reasoning can be performed through matrix computation; when used in Logic Tensor Networks (LTNs), they are able to embed first-order logic symbols such as constants, facts, and rules into real-valued tensors. The integration of KRR and ANN suggests a potential avenue for bringing biological inspiration from neuroscience into KRR. However, higher-order learning is not exclusive to human brains. Insects, such as fruit flies and honey bees, can solve simple associative learning tasks and learn abstract concepts such as "sameness" and "difference," which is viewed as a higher-order cognitive function and typically thought to depend on top-down neocortical processing. Empirical research with fruit flies strongly supports that a randomized representational architecture is used in olfactory processing in insect brains. Based on these results, we propose a Randomly Weighted Feature Network (RWFN) that incorporates randomly drawn, untrained weights in a encoder that uses an adapted linear model as a decoder. The randomized projections between input neurons and higher-order processing centers in the input brain is mimicked in RWFN by a single-hidden-layer neural network that specially structures latent representations in the hidden layer using random Fourier features that better represent complex relationships between inputs using kernel approximation. Because of this special representation, RWFNs can effectively learn the degree of relationship among inputs by training only a linear decoder model. We compare the performance of RWFNs to 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 surpass the performance of solely data-driven methods. We demonstrate that compared to LTNs, RWFNs can achieve better or similar performance for both object classification and detection of the *part-of* relations between objects in SII tasks while using much far fewer learnable parameters (1:62 ratio) and a faster learning process (1:2 ratio of running speed). Furthermore, we show that because the randomized weights do not depend on the data, several decoders can share a single randomized encoder, giving RWFNs a unique economy of spatial scale for simultaneous classification tasks.

Keywords

insect neuroscience, model architecture, randomization, neuro-symbolic computing

1. Introduction

The human brain has an extraordinary ability to memorize and learn new things to solve a variety of problems with difficulty ranging from trivial to complex. To understand the cognitive architecture of the brain, research on producing a wiring diagram of the connections among all neurons, called *Connectomics* [1], has focused not only on the human brain [2] but also on the brains of insects [3, 4], and such research has influenced the development of machine learning and Artificial Intelligence (AI) [5]. However, far more is known about the function of coarse-grained, high-level structures in the brain than the neuron-scale layout of important brain regions. Similarly, the high degrees of freedom in artificial neural networks (ANN) has provided an opportunity for the introduction of Knowledge Representation and Reasoning (KRR) to constructively constrain ANN architectures and training methods. In particular, combining KRR techniques with ANNs promises to enhance the high performance of modern AI with explainability and interpretability, which is necessary for generalized human insight and increased trustworthiness.

Several recent studies across statistical relational learning (SRL), neural-symbolic computing, knowledge completion, and approximate inference [6, 7, 8, 9] have shown that neural networks can be integrated with logical systems to perform robust learning and effective inference while also providing increased interpretability from symbolic knowledge extraction. These neural-network knowledge representation approaches use *relational embedding* to represents relational predicates in a neural network [10, 11, 12, 13]. For example, Neural Tensor Networks (NTNs) are structured to encode the degree of association among pairs of entities in the form of tensor operations on real-valued vectors [12]. These NTNs have been synthesized with neural symbolic integration [7] in the development of Logic Tensor Networks (LTNs) [14], which can extend the power of NTNs to reason over first-order many-valued logic [15].

Although KRR aims to lift the reasoning ability of computers to that of humans, such higher-order learning and reasoning capabilities are not unique to humans. Insect neuroscience has shown that insects show sophisticated and complex behaviors even though they possess miniature central nervous systems compared to the human brain [16]. For example, attention-like processes have been demonstrated in in fruit flies and honey bees [17, 18], and concept learning has been shown in bees [19]. Specifically, it has been shown that the honey bee brain contains high levels of cognitive sophistication so that it can learn relational concepts such as “same,” “different,” “larger than,” “better than,” among others, and researchers continue to study the neurobiological mechanisms and computational models underlying these capabilities [20]. Just as KRR is now being used to better shape ANN’s for more sophisticated reasoning and increased interpretability, the architectures demonstrated in the honey bee brain may provide insights into how to augment ANN’s with higher-order reasoning abilities akin to those demonstrated in insects.

NeSy’20/21 @ IJCLR: 15th International Workshop on Neural-Symbolic Learning and Reasoning, October 25–27, 2021
jhong53@asu.edu (J. Hong); tpavlic@asu.edu (T. P. Pavlic)
https://www.linkedin.com/in/jyhong0304/ (J. Hong); https://isearch.asu.edu/profile/1995237 (T. P. Pavlic)
0000-0003-4429-3311 (J. Hong); 0000-0002-7073-6932 (T. P. Pavlic)

In this paper, we propose Randomly Weighted Feature Networks (RWFNs), an insect-brain-inspired single-hidden-layer neural network for relational embedding that incorporates randomly drawn, untrained weights in its encoder with a trained linear model as a decoder. Our approach is mainly motivated by neural circuits in the insect brain centered around the Mushroom Body (MB). The MB, analogous to the neocortex in humans, is a vital region of the insect brain supporting concept learning because it is responsible for stimulus identification, categorization, and element learning [21, 22, 23]. We can model the MB as a neural network model with three layers: Input Neurons (INs) – Kenyon Cells (KCs) – mushroom body Extrinsic Neurons (ENs). One of the remarkable properties of MB is that the connections between INs and KCs are relatively random and sparse [24]. To mimic this characteristic, we used a random weight matrix to transform the input between the input and hidden layers to generate the latent representation of the relationship between real-valued input entities. By doing so, the learning process involves only the training of the weights between the hidden and output layers, which is simple and fast. In contrast, a conventional LTN would incorporate an NTN specially trained to capture logical relationships present in data, which requires more learning parameters and a more complex learning process.

Our method is also influenced by random Fourier features [25], a kernel approximation method that overcomes the issues of conventional kernel machines or kernel methods [26]. Kernel methods are one of the most powerful and theoretically grounded approaches for nonlinear statistical learning problems, including classification, regression, clustering, and others [27, 28, 29]. However, the main issue of the kernel method is the lack of scalability for large datasets and a slow training process [25, 30]. Random Fourier features can address these issues by approximating the kernel function by using an explicit feature mapping that projects the input data into a randomized feature space and by applying faster linear models to learn. Interestingly, the random Fourier features model can also be viewed as a class of a single-hidden-layer neural network model with a fixed weight between the input and hidden layers. Thus, we leveraged this to substitute the tensor operations in conventional NTNs that model the linear interactions between entities and utilized the projection from input into another space as another feature representation in the hidden layer of our model to learn relationships.

Thus, our proposed model is an insect-inspired single-hidden-layer network with latent representation derived by the integration of the input transformation between INs and KCs and random Fourier features, and it only requires training of a linear decoder. By applying the model to solve the Semantic Image Interpretation (SII) tasks, we show that a trained linear decoder in RWFNs can effectively capture the likelihood of *part-of* relationships at a level of performance exceeding that of traditional LTNs, even with far fewer parameters and a faster learning process. To the best of our knowledge, this is the first research to integrate both insect neuroscience and neuro-symbolic approaches for reasoning under uncertainty and for learning in the presence of data and rich knowledge. Furthermore, because the encoder weights in our model do not depend upon the data, the single encoder can be shared among several decoders, each trained for a different classifier, giving RWFNs an economy of spatial scale in our model applications where several classifiers need to be used simultaneously.

2. Related Work and Background

Insect Neuroscience The MB in the insect brain receives processed olfactory, visual, and mechanosensory stimuli [31] and is viewed as the critical region responsible for multimodal associative learning [21]. In the fruit fly, thousands of Kenyon Cells (KCs) in the MB each receive a set of random ~7 inputs from INs [24, 32], and this is similarly true for honey bees [33]. A simplified neural circuit modeling the MB is a neural network with three layers consisting of: i) INs that provide olfactory, visual, and mechanosensory inputs, ii) KCs generating the sparse-encoding of sensory stimuli, and iii) ENs for activating several different behavioral responses [34]. In particular, INs receive various inputs from Antennal Lobe (AL) glomeruli, Medulla, and Lobula optic neuropils [34]. For simplicity, we focus on the olfactory pathway between glomeruli in the AL and KCs in the MB [34, 33]. The insect olfactory neural circuit has a divergence–convergence structure where ~800 AL glomeruli form a coded feature vector that expands into a spare representation across ~170,000 KCs, and these are decoded by ~400 ENs that actuate motor pathways based on this information-processing pipeline [33]. This general divergence–convergence structure applies equally as well to honey bees and fruit flies [35]; therefore, for modeling the architecture of our methods for brevity, we interchangeably leverage the neural circuits of olfactory nervous systems between these insects.

In this paper, the input transformation of odorant representation in the AL to the higher-order representation across the KCs in the MB is shown in the projection between the input and hidden layers of our model, and this representation plays the critical role in learning the relationships presented on the input.

Random Fourier Features Kernel machines, e.g., Support Vector Machines (SVMs) [36, 37], have received significant attention due to their capability for function approximation and excellent performance of detecting decision boundaries with enough training data. These methods use transformations, as with a lifting function ϕ, that help to better discriminate among different inputs. Given dataset vector inputs $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, the kernel function $k(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle$ represents the similarity (i.e., inner product) between $\mathbf{x}$ and $\mathbf{y}$ in the transformed space. However, because of the potential complexity of the transformation ϕ, learning the kernel function k may require significant computational and storage costs.

Random Fourier features [25], instead, provide a data transformation that permits using a far less expensive approximation of the kernel function. For each vector input $\mathbf{x} \in \mathbb{R}^d$, the technique applies a randomized feature function $\mathbf{z} : \mathbb{R}^d \to \mathbb{R}^D$ (generally, $D \gg d$ with sample size $N \gg D$) that maps $\mathbf{x}$ to evaluations of D random Fourier basis from the Fourier transform of kernel k. In this transformed space, kernel evaluations can be approximated by linear operations, as in:

$$k(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle \approx \mathbf{z}(\mathbf{x})^\top \mathbf{z}(\mathbf{y}) \tag{1}$$

Thus, by transforming the input with $\mathbf{z}$, fast linear learning methods can be leveraged to approximate the evaluations of nonlinear kernel machines.

In this paper, we use random Fourier features as latent representations that reduce the complexity of learning relations among real-valued entities. As described in Section 1, adaptable NTNs within LTNs have been used to encode relationships among real-valued entities. We replace the adaptable NTNs with random Fourier features that have high expressiveness with low decoding overhead. Details and intuitions will be given in Section 3.

Logic Tensor Networks (LTNs) The RWFNs we propose are meant to improve upon LTNs for statistical relational learning tasks. LTNs integrate learning based on NTNs [12] with reasoning using first-order, many-valued logic [15], all implemented in TensorFlow [14]. Here, we briefly introduce LTN syntax and semantics for use in mapping logical symbols to numerical values and learning reasoning relations among real-valued vectors using the logical formulas.

Although a first-order-logic (FOL) language $\mathscr{L}$ and its signature are defined by consisting of three disjoint sets – i) $\mathscr{C}$ (constants), ii) $\mathscr{F}$ (functions) and iii) $\mathscr{P}$(predicate) – we ignore function symbols $\mathscr{F}$ because they are not used in SII tasks that we focus on here. For any predicate symbol s, $\alpha(s)$ can be described as its *arity*, and logical formulas in $\mathscr{L}$ enable the description of relational knowledge. The objects being reasoned over with FOL are mapped to an interpretation domain $\subseteq \mathbb{R}^n$ 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, 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.

Let $n \in \mathbb{N}$. An n-grounding, or simply grounding, $\mathscr{G}$ for a FOL $\mathscr{L}$ is a function defined on the signature of $\mathscr{L}$ satisfying the following conditions:

$$\mathscr{G}(c) \in \mathbb{R}^n \text{ for every constant symbol } c \in \mathscr{C}$$
$$\mathscr{G}(P) \in \mathbb{R}^{n \cdot \alpha(f)} \to [0,1] \text{ for predicate sym. } P \in \mathscr{P}$$

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

$$\mathscr{G}(P(t_1, \dots, t_m)) \triangleq \mathscr{G}(P)(\mathscr{G}(t_1), \dots, \mathscr{G}(t_m))$$

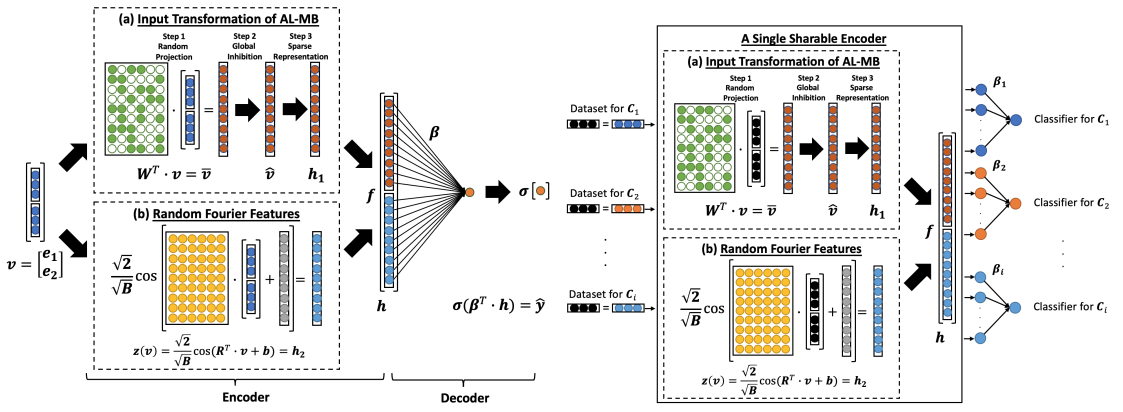
The semantics for connectives, such as $\mathscr{G}(\neg\phi), \mathscr{G}(\phi \wedge \psi), \mathscr{G}(\phi \vee \psi)$, and $\mathscr{G}(\phi \to \psi)$, can be computed by following the fuzzy logic such as the Lukasiewicz t-norm [15].

A partial grounding $\hat{\mathscr{G}}$ can be defined on a subset of the signature of $\mathscr{L}$. A grounding $\mathscr{G}$ is said to be a completion of $\hat{\mathscr{G}}$ if $\mathscr{G}$ is a grounding for $\mathscr{L}$ and coincides with $\hat{\mathscr{G}}$ on the symbols where $\hat{\mathscr{G}}$ is defined. Let GT be a grounded theory which is a pair $\langle \mathscr{K}, \hat{\mathscr{G}} \rangle$ with a set $\mathscr{K}$ of closed formulas and a partial grounding $\hat{\mathscr{G}}$. A grounding $\mathscr{G}$ satisfies a GT $\langle \mathscr{K}, \hat{\mathscr{G}} \rangle$ if $\mathscr{G}$ completes $\hat{\mathscr{G}}$ and $\mathscr{G}(\phi) = 1$ for all $\phi \in \mathscr{K}$. A GT $\langle \mathscr{K}, \hat{\mathscr{G}} \rangle$ is satisfiable if there exists a grounding $\mathscr{G}$ that satisfies $\langle \mathscr{K}, \hat{\mathscr{G}} \rangle$. In other words, deciding the satisfiability of $\langle \mathscr{K}, \hat{\mathscr{G}} \rangle$ amounts to searching for a grounding $\mathscr{G}$ such that all the formulas of $\mathscr{K}$ are mapped to 1. If a GT is not satisfiable, the best possible satisfaction that we can reach with a grounding is of our interest.

Grounding $\mathscr{G}^*$ captures the implicit correlation between quantitative features of objects and their categorical/relational properties. The grounding of an m-ary predicate P, namely $\mathscr{G}(P)$, is defined as a generalization of the NTN [12], as a function from $\mathbb{R}^{mn}$ to $[0,1]$, as follows:

$$\mathscr{G}_{LTN}(P)(\mathbf{v}) = \sigma(u_P^\top \mathrm{f}(\mathbf{v}^\top W_P^{[1:k]}\mathbf{v} + V_P\mathbf{v} + b_P)) \tag{2}$$

where $\mathbf{v} = \langle \mathbf{v}_1^\top, \dots, \mathbf{v}_m^\top \rangle^\top$ is the mn-ary vector obtained by concatenating each $\mathbf{v}_i$. σ is the sigmoid function and f is the hyperbolic tangent (tanh). The parameters for P are: $W_P^{[1:k]}$, a 3-D tensor


(a) Input Transformation of AL-MB
Step 1 Random Projection
Step 2 Global Inhibition
Step 3 Sparse Representation
$W^T \cdot v = \bar{v}$
$\hat{v}$
h_1
(b) Random Fourier Features
$\frac{\sqrt{2}}{\sqrt{B}}\cos$
$z(v) = \frac{\sqrt{2}}{\sqrt{B}}\cos(R^T \cdot v + b) = h_2$
$v = \begin{bmatrix} e_1 \\ e_2 \end{bmatrix}$
f
β
h
$\sigma[\,]$
$\sigma(\beta^T \cdot h) = \hat{y}$
Encoder
Decoder
A Single Sharable Encoder
Dataset for C_1
Dataset for C_2
Dataset for C_i
β_1
β_2
β_i
Classifier for C_1
Classifier for C_2
Classifier for C_i

(a) Visualization of the structure of the Randomly Weighted Feature Network. In the depicted case, the input vector $\mathbf{v}$ constitutes of two entities, $e_1, e_2 \in \mathbb{R}^3$ and it shows to learn a binary relation between them (e_1, R, e_2), such as (Cat, hasPart, Tail).

(b) Visualization of the structure of the Randomly Weighted Feature Network with weight sharing. In the case of learning each classifier from the class $\mathscr{C}_1$ to the class $\mathscr{C}_i$, RWFNs allow us to use the same encoder to extract features from each data from the class $\mathscr{C}_1$ to the class $\mathscr{C}_i$.

Figure 1: The architectures of RWFNs and RWFNs with weight sharing

in $\mathbb{R}^{k\times mn\times mn}$, $V_P \in \mathbb{R}^{k\times mn}$, $b_P \in \mathbb{R}^k$ and $u_P \in \mathbb{R}^k$. Because our RWFN model can be used to ground a predicate as $\mathscr{G}_{RWFN}(P)$, we can directly compare the performance of RWFNs for the SII tasks with LTNs.

3. Randomly Weighted Feature Networks (RWFNs)

In this section, we introduce the details of Randomly Weighted Feature Networks (RWFNs). The underlying intuition behind the development of this model can be found in Appendix A.

3.1. Model Architecture

Let the input vector $\mathbf{v}$ be $[\mathbf{v}_1^\top, \ldots, \mathbf{v}_m^\top]^\top$, the mn-ary vector where m is arity and n is the input dimension. We first define the two kinds of latent representations: i) the input transformation between AL glomeruli and KCs inspired by the insect brain, and ii) the transformed input using a randomized feature mapping $\mathbf{z}(\cdot)$ in random Fourier features.

For the bio-inspired representation, we select $N_{in} \in [1, mn)$ indices of the input at random without replacement for each hidden node[1]. In other words, the output $\bar{v}_j$ of each hidden node $j \in \{1, ..., B\}$ is a weighted combination of all mn inputs where only $N_{in} < mn$ inputs have $w_{j,i} = 1$ and all other inputs have $w_{j,i} = 0$. So the inputs are effectively gated by the weights on each hidden node and the weight matrix $\mathbf{W} \in \mathbb{R}^{mn\times B}$ in this computation is random, binary, and sparse.

[1]$N_{in} < mn$ to prevent hidden-unit outputs becoming trivially 0 by Eq. (3). In our setting, $N_{in} = 7$.

Once the jth hidden node has produced weighted sum $\bar{v}_j$, the post-processing step to produce intermediate output is performed in the hidden layer. Mimicking Eq. (6) with $C = 1$, the jth intermediate output $\hat{x}_j$ is:

$$\hat{v}_j = \bar{v}_j - \mu, \quad \mu = \frac{1}{B}\sum_{i=1}^{B} \bar{v}_i \tag{3}$$

where B is the number of hidden units. Therefore, the sparse output of the j-th KC node can be defined as $h_j^{(1)} = g(\hat{v}_j)$ where g is the ReLU function [38] that allows the model to produce sparse hidden output, which is more biologically plausible. By doing so, we define the output vector as $\mathbf{h}_1 = [h_1^{(1)}, \dots, h_B^{(1)}]^\top$.

On the other hand, to generate random Fourier features, we used a randomized feature function $\mathbf{z}(\cdot)$ in [25, 39], we can project the input as follows:

$$\mathbf{h}_2 = \mathbf{z}(\mathbf{v}) = \frac{\sqrt{2}}{\sqrt{B}} \cos(\mathbf{R}^\top \mathbf{v} + \mathbf{b}) \tag{4}$$

where $\mathbf{R} \sim Normal^{mn \times B}(0, 1)$ and $\mathbf{b} \sim Uniform^{B}(0, 2\pi)$, which is Gaussian kernel approximation. Consequently, the output vector $\mathbf{h}_2$ can be considered as another latent representation of relationship among input. Rahimi et al. [25], Sutherland and Schneider [39], and Liu et al. [30] provide theoretical derivations of kernel approximation and comparative analyses of various kinds of random Fourier features.

Finally, using the above two latent representations, our RWFNs can be defined as a function from $\mathbb{R}^{mn}$ to $[0, 1]$:

$$\mathcal{G}_{RWFN}(P)(\mathbf{v}) = \sigma\left(\boldsymbol{\beta}^\top \mathbf{h}\right) = \sigma\left(\boldsymbol{\beta}^\top \mathrm{f}\left(\begin{bmatrix}\mathbf{h}_1 \\ \mathbf{h}_2\end{bmatrix}\right)\right) \tag{5}$$

where $\mathbf{h}$ is the final hidden representation obtained by applying the hyperbolic tangent (tanh) function f to the concatenation of $\mathbf{h}_1$ and $\mathbf{h}_2$, and σ is the sigmoid function; the tanh function was used for the numeric stabilization. Because our model requires to adapt only $\boldsymbol{\beta} \in \mathbb{R}^{2B}$, it possess a faster learning process with fewer parameters compared to LTNs. Fig. 1a shows a visualization of the structure of our model.

3.2. RWFNs with Weight Sharing

In the insect brain, extrinsic neurons from the MB are processed by several small, downstream neuropils that ultimately lead to decision-making outcomes, such as muscle actuation. If we view these small neuropils as decoding the complex representations in the MB, then different decoders responsible for different decisions all use information sourced from the same randomized representations in the MB. The MB can be viewed as a generalized encoder that is not tailored for a particular task; consequently, it provides a shared resource to reduce the complexity of these downstream neuropils.

Because the weights of the randomized encoder of an RWFN are independent of the training data, they can also serve as a shared resource for multiple relatively simple (i.e., linear) downstream decoders trained for different classifiers. We refer to this property as *weight sharing*. Fig. 1b shows a visualization of the structure of our model applied with weight sharing to the learning of *i* different classifiers. The large solid box surrounds a single encoder that serves as a common feature extractor for all classifiers.

The entities in the original definition of RWFNs in Fig. 1a become the placeholder to be injected by the input for each classifier. Instead of generating the randomized encoder for each classifier, each classifier uses the same encoder, and training only requires learning the weights of that classifier's highly simple linear decoder. This approach increases reusability and cost efficiency in a way beyond what is possible with LTNs, which must train all encoder and decoder networks separately for each classifier.

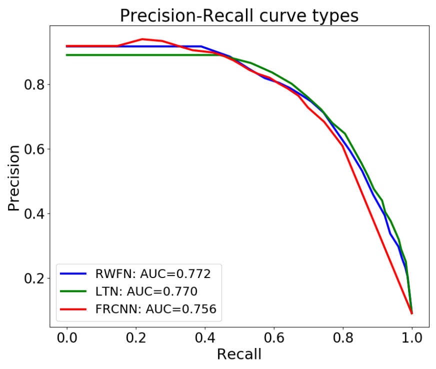
4. Experimental Evaluation

To evaluate the performance of our proposed RWFNs over LTNs, we employ both for 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. [40] 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) [41]. Our experiments are conducted by comparing the performance of two tasks of SII between RWFNs and LTNs. These tasks are well defined in first-order logic, and the codes implemented in TensorFlow framework have been provided and can be used to compare the performance of LTNs with RWFNs.

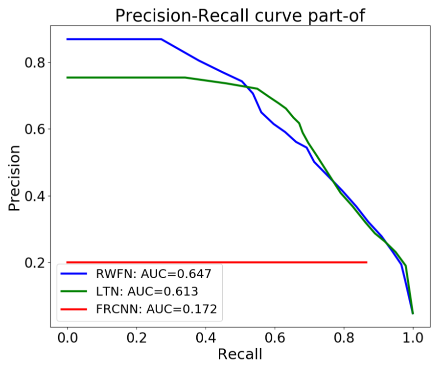
4.1. Methods

Here, we provide details of our experimental comparison of RWFNs and LTNs. We utilize the formalization of SII in first-order logic from Donadello et al. [40]. For brevity, we describe: (i) the difference of the ground theories between RWFNs and LTNs, (ii) the data set used in the experiments (Appendix B), and (iii) the RWFN and LTN hyperparameters used in the experiments (Appendix B). We omit other formalization details of the SII tasks that can be found elsewhere [40].

Defining the Grounded Theories for RWFNs and LTNs A set of bounding boxes of images correctly labelled with the classes that they belong to and pairs of bounding boxes that properly labelled with the *part-of* relation were provided. These datasets can be considered as a training set, and a grounded theory $\mathcal{T}_{\text{LTN}} \triangleq \langle \mathcal{K}, \hat{\mathcal{G}}_{LTN} \rangle$ can be constructed. In particular, $\mathcal{K}$ contains: (i) the set of closed literals $C_i(b)$ and $\texttt{partOf}(b, b')$ for every bounding box b labelled with C_i and for every pair of bounding boxes $\langle b, b' \rangle$ connected by the $\texttt{partOf}$ relation, and (ii) the set of the mereological constraints for the *part-of* relation, including asymmetric constraints, lists of several parts of an object, or restrictions that whole objects cannot be part of other objects and every part object cannot be divided further into parts. Furthermore, the partial grounding $\hat{\mathcal{G}}_{LTN}$ is defined on all bounding boxes of all the images in the training set where both $class(C_i, b)$ and the bounding box coordinates are computed by the Fast R-CNN object detector. $\hat{\mathcal{G}}$ is not defined for the predicate symbols in $\mathcal{P}$and is to be learned.



(a) RWFNs achieve similar performance for object type classification compared to LTNs, achieving an Area Under the Curve (AUC) of 0.772 (compared to 0.770).



(b) RWTNs outperform LTNs on the detection of *part-of* relations, achieving AUC of 0.647 (compared to 0.613).

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

A grounded theory $\mathcal{T}_{\text{RWTN}} \triangleq \langle \mathcal{K}, \hat{\mathcal{G}}_{RWFN} \rangle$ where a partial grounding $\hat{\mathcal{G}}_{RWFN}$ can be described for predicates using Eq. (5). Thus, we can easily compare the performance between $\hat{\mathcal{G}}_{RWFN}$ (Eq. (5)) and $\hat{\mathcal{G}}_{LTN}$ (Eq. (2)).

4.2. Results

Our experiments mainly focus on the comparison of the performance between our model and LTN, but figures also include results with Fast-RCNN [41] for type classification and the inclusion ratio *ir* baseline in 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 `partOf` relation. Every bounding box b is classified into $C \in \mathcal{P}_1$ if $\mathcal{G}(C(b)) > th$.

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 and object types classification, RWFNs achieve better performance than LTNs. However, there is 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 our model can achieve similar performance as LTNs for object-task classification and superior performance for detection of *part-of* relations.

In Table 1, we only included AUC numbers for RWFNs with weight sharing (third column) for object-type classification because *part-of* relations only require a single classifier. The performance of RWFNs with weight sharing for the object-type classification task (which requires 11 classifiers for indoor objects, 23 for vehicles, and 26 for animals) shows only a marginal gap in performance compared to other models, which demonstrates the effectiveness and efficiency of the approach of using a single shared encoder in RWFNs with weight sharing.

As summarized in Appendix C, we also conducted ablation studies to assess the degree to

Table 1
AUC of T1 (object type classification) and T2 (detection of *part-of* relation) for LTN, RWFN, and RWFN with weight sharing across label groups. $\text{MEAN}_{\pm 2\times \text{SD}}$ for all models. Best performances shown in **bold**.

Label-Task	LTN	RWFN	RWFN w/ W.S
Indoor-T1	$.769_{\pm.0314}$	$.770_{\pm.0092}$	$\mathbf{.773}_{\pm.028}$
Indoor-T2	$.619_{\pm.082}$	$\mathbf{.648}_{\pm.0621}$	—
Vehicle-T1	$.709_{\pm.0289}$	$\mathbf{.711}_{\pm.0162}$	$.706_{\pm.0111}$
Vehicle-T2	$.576_{\pm.0355}$	$\mathbf{.613}_{\pm.0489}$	—
Animal-T1	$\mathbf{.701}_{\pm.024}$	$.700_{\pm.024}$	$.697_{\pm.0237}$
Animal-T2	$.640_{\pm.0783}$	$\mathbf{.661}_{\pm.0364}$	—

which the AL–MB input transformation and the random Fourier features each contribute to the performance of the model. We have also included, in Appendix D, a detailed comparison of performance among LTNs, RWFNs, and RWFNs with weight sharing. Specifically, we compare LTNs and RWFNs in terms of numbers of learnable parameters and running times; we also compare RWFNs with and without weight sharing in terms of space complexity.

5. Conclusion and Future Work

In this paper, we introduced Randomly Weighted Feature Networks, which incorporate the insect-brain-inspired neuronal feature representation and unique random features derived by random Fourier features. The RWFN encoder acts as a generalized feature extractor with greater relational expressiveness and a learning model with relatively simpler structure. We demonstrated how insights from the insect nervous system can be applied to the fields of neural-symbolic computing and knowledge representation and reasoning for relational learning.

Our work can be advanced in several ways. For one, RWFNs can be applied to other variants of SII problems proposed by Donadello and Serafini [42], and performance between our model and LTN for zero-shot learning in SII tasks can be compared. In addition, we plan to extend application of RWFNs to tasks that need to extract structural knowledge from not only images but also text, such as visual question-answering challenges. Furthermore, we will investigate how other methods from neuroscience for exploring biologically-plausible learning algorithms might be applicable to our model. Finally, we will extend RWFNs to include a recurrent part for representing dynamic features of time-series data, similar to reservoir computing [43, 44, 45]; this approach may allow for extracting time-varying relational knowledge necessary for developing a framework for data-driven reasoning over temporal logic.

Acknowledgments

This work was supported in part by NSF SES-1735579.

References

[1] S. Seung, Connectome: How the brain's wiring makes us who we are, HMH, 2012.
[2] O. Sporns, G. Tononi, R. Kötter, The human connectome: a structural description of the human brain, PLoS Comput Biol 1 (2005) e42.
[3] K. Eichler, F. Li, A. Litwin-Kumar, Y. Park, I. Andrade, C. M. Schneider-Mizell, T. Saumweber, A. Huser, C. Eschbach, B. Gerber, et al., The complete connectome of a learning and memory centre in an insect brain, Nature 548 (2017) 175–182.
[4] S.-y. Takemura, Y. Aso, T. Hige, A. Wong, Z. Lu, C. S. Xu, P. K. Rivlin, H. Hess, T. Zhao, T. Parag, et al., A connectome of a learning and memory center in the adult drosophila brain, Elife 6 (2017) e26975.
[5] D. Hassabis, D. Kumaran, C. Summerfield, M. Botvinick, Neuroscience-inspired artificial intelligence, Neuron 95 (2017) 245–258.
[6] D. Koller, N. Friedman, S. Džeroski, C. Sutton, A. McCallum, A. Pfeffer, P. Abbeel, M.-F. Wong, D. Heckerman, C. Meek, et al., Introduction to statistical relational learning, MIT press, 2007.
[7] A. S. Garcez, L. C. Lamb, D. M. Gabbay, Neural-symbolic cognitive reasoning, Springer Science & Business Media, 2008.
[8] J. Pearl, Probabilistic reasoning in intelligent systems: networks of plausible inference, Elsevier, 2014.
[9] M. Nickel, K. Murphy, V. Tresp, E. Gabrilovich, A review of relational machine learning for knowledge graphs, Proceedings of the IEEE 104 (2015) 11–33.
[10] I. Sutskever, G. E. Hinton, Using matrices to model symbolic relationship, in: Advances in neural information processing systems, 2009, pp. 1593–1600.
[11] A. Bordes, J. Weston, R. Collobert, Y. Bengio, Learning structured embeddings of knowledge bases, in: Twenty-Fifth AAAI Conference on Artificial Intelligence, 2011.
[12] R. Socher, D. Chen, C. D. Manning, A. Ng, Reasoning with neural tensor networks for knowledge base completion, in: Advances in neural information processing systems, 2013, pp. 926–934.
[13] A. Santoro, D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, T. Lillicrap, A simple neural network module for relational reasoning, in: Advances in neural information processing systems, 2017, pp. 4967–4976.
[14] L. Serafini, A. d. Garcez, Logic tensor networks: Deep learning and logical reasoning from data and knowledge, arXiv preprint arXiv:1606.04422 (2016).
[15] M. Bergmann, An introduction to many-valued and fuzzy logic: semantics, algebras, and derivation systems, Cambridge University Press, 2008.
[16] A. Avarguès-Weber, N. Deisig, M. Giurfa, Visual cognition in social insects, Annual review of entomology 56 (2011) 423–443.
[17] B. van Swinderen, R. J. Greenspan, Salience modulates 20–30 hz brain activity in drosophila, Nature neuroscience 6 (2003) 579–586.
[18] J. Spaethe, J. Tautz, L. Chittka, Do honeybees detect colour targets using serial or parallel visual search?, Journal of Experimental Biology 209 (2006) 987–993.
[19] A. Avarguès-Weber, A. G. Dyer, M. Combe, M. Giurfa, Simultaneous mastering of two abstract concepts by the miniature brain of bees, Proceedings of the National Academy of

Sciences 109 (2012) 7481–7486.
[20] A. Avarguès-Weber, M. Giurfa, Conceptual learning by miniature brains, Proceedings of the Royal Society B: Biological Sciences 280 (2013) 20131907.
[21] R. Menzel, Searching for the memory trace in a mini-brain, the honeybee, Learning & memory 8 (2001) 53–62.
[22] C. G. Galizia, Olfactory coding in the insect brain: data and conjectures, European Journal of Neuroscience 39 (2014) 1784–1795.
[23] M. Bazhenov, R. Huerta, B. H. Smith, A computational framework for understanding decision making through integration of basic learning rules, Journal of Neuroscience 33 (2013) 5686–5697.
[24] S. J. Caron, V. Ruta, L. Abbott, R. Axel, Random convergence of olfactory inputs in the drosophila mushroom body, Nature 497 (2013) 113–117.
[25] A. Rahimi, B. Recht, et al., Random features for large-scale kernel machines., in: NIPS, volume 3, Citeseer, 2007, p. 5.
[26] A. J. Smola, B. Schölkopf, Learning with kernels, volume 4, Citeseer, 1998.
[27] J. Zhu, T. Hastie, Kernel logistic regression and the import vector machine, Journal of Computational and Graphical Statistics 14 (2005) 185–205.
[28] H. Drucker, C. J. Burges, L. Kaufman, A. Smola, V. Vapnik, et al., Support vector regression machines, Advances in neural information processing systems 9 (1997) 155–161.
[29] I. S. Dhillon, Y. Guan, B. Kulis, Kernel k-means: spectral clustering and normalized cuts, in: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, 2004, pp. 551–556.
[30] F. Liu, X. Huang, Y. Chen, J. A. Suykens, Random features for kernel approximation: A survey in algorithms, theory, and beyond, arXiv preprint arXiv:2004.11154 (2020).
[31] P. Mobbs, The brain of the honeybee apis mellifera. i. the connections and spatial organization of the mushroom bodies, Philosophical Transactions of the Royal Society of London. B, Biological Sciences 298 (1982) 309–354.
[32] K. Inada, Y. Tsuchimoto, H. Kazama, Origins of cell-type-specific olfactory processing in the drosophila mushroom body circuit, Neuron 95 (2017) 357–367.
[33] F. Peng, L. Chittka, A simple computational model of the bee mushroom body can explain seemingly complex forms of olfactory learning and memory, Current Biology 27 (2017) 224–230.
[34] A. J. Cope, E. Vasilaki, D. Minors, C. Sabo, J. A. Marshall, A. B. Barron, Abstract concept learning in a simple neural network inspired by the insect brain, PLoS computational biology 14 (2018) e1006435.
[35] K. Endo, Y. Tsuchimoto, H. Kazama, Synthesis of conserved odor object representations in a random, divergent-convergent network, Neuron 108 (2020) 367–381.
[36] B. E. Boser, I. M. Guyon, V. N. Vapnik, A training algorithm for optimal margin classifiers, in: Proceedings of the fifth annual workshop on Computational learning theory, 1992, pp. 144–152.
[37] C. Cortes, V. Vapnik, Support-vector networks, Machine learning 20 (1995) 273–297.
[38] X. Glorot, A. Bordes, Y. Bengio, Deep sparse rectifier neural networks, in: Proceedings of the fourteenth international conference on artificial intelligence and statistics, JMLR Workshop and Conference Proceedings, 2011, pp. 315–323.

[39] D. J. Sutherland, J. Schneider, On the error of random fourier features, arXiv preprint arXiv:1506.02785 (2015).
[40] I. Donadello, L. Serafini, A. D. Garcez, Logic tensor networks for semantic image interpretation, arXiv preprint arXiv:1705.08968 (2017).
[41] R. Girshick, Fast r-cnn, in: Proceedings of the IEEE international conference on computer vision, 2015, pp. 1440–1448.
[42] I. Donadello, L. Serafini, Compensating supervision incompleteness with prior knowledge in semantic image interpretation, in: 2019 International Joint Conference on Neural Networks (IJCNN), IEEE, 2019, pp. 1–8.
[43] A. A. Ferreira, T. B. Ludermir, Genetic algorithm for reservoir computing optimization, in: 2009 International Joint Conference on Neural Networks, IEEE, 2009, pp. 811–815.
[44] X. Sun, T. Li, Q. Li, Y. Huang, Y. Li, Deep belief echo-state network and its application to time series prediction, Knowledge-Based Systems 130 (2017) 17–29.
[45] X. Wang, Y. Jin, K. Hao, Echo state networks regulated by local intrinsic plasticity rules for regression, Neurocomputing 351 (2019) 111–122.
[46] S. Dasgupta, C. F. Stevens, S. Navlakha, A neural algorithm for a fundamental computing problem, Science 358 (2017) 793–796.
[47] Y. Aso, K. Grübel, S. Busch, A. B. Friedrich, I. Siwanowicz, H. Tanimoto, The mushroom body of adult drosophila characterized by gal4 drivers, Journal of neurogenetics 23 (2009) 156–172.
[48] X. Chen, R. Mottaghi, X. Liu, S. Fidler, R. Urtasun, A. Yuille, 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, 2014, pp. 1971–1978.
[49] T. Tieleman, G. Hinton, Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude, COURSERA: Neural Networks for Machine Learning, 2012.
[50] T. Akiba, S. Sano, T. Yanase, T. Ohta, M. Koyama, Optuna: A next-generation hyperparameter optimization framework, in: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, 2019, pp. 2623–2631.

A. The Intuitions of RWFNs

For the bio-inspired representation in our model, we concentrated on implementing: (i) how to build random connections between the AL glomeruli (input layer) and the KCs (hidden layer), and (ii) how to guarantee hidden-layer sparsity to best differentiate one odor stimulus from another. We used a sparse, binary, and random matrix to define an arbitrary set of inputs for each KC in the model with inspiration from Caron et al. [24], Peng and Chittka [33], and Dasgupta et al. [46]. In particular, in biological models of the insect brain and the AL–MB interface, the firing rates from 7 randomly selected glomeruli are passed and summed to each KC [24, 32]. Furthermore, Endo et al. [35] developed a computational model of sparsity of the KCs' output activity based on global inhibition the average KC input. In their model, KCs output an intermediate result subject to global inhibition from the average glomerular input to all KCs. The last KC activity is then produced by thresholding the inhibited output through a ramp function, which is functionally equivalent to the Rectified Linear Unit (ReLU) activation function [38]. Thus, the output of the jth KC, KC_{out_j}, in the computational model was described as:

$$KC_{out_j} = \phi\left(KC_{in_j} - C\frac{1}{N_{KC}}\sum_{j}^{N_{KC}} KC_{in_j}\right) \tag{6}$$

where KC_{in_j} indicates the weighted sum of input from 7 random indices of the input vector, ϕ is the ReLU, C is the strength of global inhibition, and N_{KC} is the total number of KCs. The parameters $C = 1.0$ and $N_{KC} = 2000$ were chosen so as to match the values best calibrated to real KC responses [35, 47]. With this KC representation, Endo et al. [35] trained a linear decoder to successfully classify 'group' from 'non-group' odors. Similarly, we make use of Eq. (6) and train a linear model for learning latent relationships among input.

For another hidden representation using random Fourier features in our model, based on Eq. (1), we can define a decision function $f(\mathbf{x})$ given a dataset including N data samples $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ and a randomized feature mapping $\mathbf{z} : \mathbb{R}^d \to \mathbb{R}^D$ as follows:

$$\begin{aligned} f(\mathbf{x}) &= \sum_{n=1}^{N} \alpha_n k(\mathbf{x}_n, \mathbf{x}) = \sum_{n=1}^{N} \alpha_n \langle \phi(\mathbf{x}_n), \phi(\mathbf{x}) \rangle \\ &\approx \sum_{n=1}^{N} \alpha_n \mathbf{z}(\mathbf{x}_n)^\top \mathbf{z}(\mathbf{x}) = \boldsymbol{\beta}^\top \mathbf{z}(\mathbf{x}) \end{aligned} \tag{7}$$

This indicates that if $\mathbf{z}(\cdot)$ can approximate $\phi(\cdot)$ well, we can simply map our data using $\mathbf{z}(\cdot)$ and then use a linear model to learn because both $\boldsymbol{\beta}$ and $\mathbf{z}(\cdot)$ in the above equation are D-vectors. Therefore, the task that we will describe in the next section is how to find a random projection function $\mathbf{z}(\cdot)$ that can approximate the corresponding nonlinear kernel machine appropriately.

The reason why we leverage the random Fourier feature is conciseness and efficiency of computing linear interactions among input, which can be replaced with the bilinear model in Eq. (2). In Eq. (2), the bilinear tensor was used to compute the relation, which seems intuitive because each slice of the tensor serves as being responsible for one type of relation. However, this computation requires high computational cost with large number of parameters. In contrast, the random Fourier features in Eq (7) can do the similar task with a much faster learning process and fewer number of parameters.

Considering how Eq. (6) and Eq. (7) can be used in our model, the hidden representation can be expressed with the concatenation of KC_{out} and $\mathbf{z}(\cdot)$.

B. Details of Experiments

Hardware specification of the server The hardware specification of the server that we used to experiment is as follows:

- CPU: Intel® Core$^{\text{TM}}$ i7-6950X CPU @ 3.00GHz (up to 3.50 GHz)
- RAM: 128 GB (DDR4 2400MHz)
- GPU: NVIDIA GeForce Titan Xp GP102 (Pascal architecture, 3840 CUDA Cores @ 1.6 GHz, 384 bit bus width, 12 GB GDDR G5X memory)

Source codes All source codes, trained models, and figures in this paper are available at https://github.com/jyhong0304/SII.

Datasets The PASCAL-Part-dataset [48] and ontologies (WordNet) are chosen for the *part-of* 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 59 labels (20 labels for whole objects and 39 labels for parts). 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 *part-of* 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.

Hyperparameter Setting To compare the performance between RWFNs and LTNs, we trained two models separately. For LTN, we configure the experimental environment following Donadello et al. [40]. The LTNs were configured with a tensor of $k = 6$ layers. For RWFN, the number of hidden nodes $B = 200$ for a classifier for object type classification. In addition, we set the number of hidden nodes of a classifier for *part-of* detection as twice as large as the number of hidden nodes B of a classifier for object classification, which is 400. This is because the dimension of inputs that the classifier for detecting the *part-of* relation is twice as large as the input space required for a classifier for objection categorization. Referring to Donadello et al. [40], both models make use of a regularization parameter $\lambda = 10^{-10}$, Lukasiewicz's *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 [49] learning algorithm available in TensorFlow for each model.

Hyperparameter Searching for RWFNs To find out the best number of hidden nodes B, we used the Optuna framework [50] with 500 iterations in the range of $[64, 512]$. The Optuna framework allows us to dynamically construct the parameter search space because we can formulate hyperparameter optimization as the maximization/minimization process of an objective function that takes a set of hyperparameters as input and returns a validation score. In our case, the validation score returned was the test AUC values. Furthermore, it provides efficient sampling methods, such as relational sampling that exploits the correlations among the parameters.

C. Ablation Studies

Table 2 shows the results of ablation studies. In order to show how much two hidden representations – the input transformation between AL–MB and random Fourier features – contribute to the performance of our model, we built two separate RWFN models: one using the AL–MB input transformation only and another using random Fourier features only. Then, we performed

Table 2
AUC of T1 (object type classification) and T2 (detection of *part-of* relation) for the AL–MB representation (AL–MB) and random Fourier features (RFF) across label groups. MEAN$_{\pm 2\times \text{SD}}$ for all models. The best performance is displayed in bold.

Label-Task	AL–MB	RFF
Indoor-T1	$.743_{\pm.021}$	$\mathbf{.766}_{\pm.012}$
Indoor-T2	$.525_{\pm.102}$	$\mathbf{.641}_{\pm.010}$
Vehicle-T1	$.710_{\pm.017}$	$\mathbf{.715}_{\pm.009}$
Vehicle-T2	$\mathbf{.612}_{\pm.027}$	$.572_{\pm.079}$
Animal-T1	$.705_{\pm.017}$	$\mathbf{.709}_{\pm.013}$
Animal-T2	$\mathbf{.664}_{\pm.069}$	$.646_{\pm.020}$

five experiments and averaged AUCs of each model for object classification and *part-of* detection. The number of hyperparameter β for each model was set to the same as the number of hyperparameter for the original RWFN, which is 200.

For the object-type classification and *part-of* detection tasks using Indoor label, the random Fourier features outperform the AL–MB input transformation. On the other hand, the AL–MB input transformation for *part-of* detection tasks using Vehicle and Animal labels show better performance than the random Fourier features. Therefore, these ablation studies show that the model architecture of RWFNs in Eq. (5) can fully utilize both hidden representations and contribute to their good performance shown in Table. 1 by compensating for each other.

D. Performance Analysis

Relative Complexity of RWFNs and LTNs To better appreciate the relative performance of RWFNs and LTNs, we can compare the number of parameters for grounding a unary predicate for each model. The dimension of the input in the dataset for both RWFNs and LTNs is $n = 64$. As shown in Eq. (2), 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 $(n^2 + n + 2) \cdot k = (64^2 + 64 + 2) \cdot 6 = 24972$. On the other hand, in Eq. (3) and Eq. (4), the number of parameters in RWFNs are $\{\mathbf{W} \in \mathbb{R}^{n\times B}, \mathbf{R} \in \mathbb{R}^{n\times B}, \mathbf{b} \in \mathbb{R}^B, \boldsymbol{\beta} \in \mathbb{R}^{2B}\}$, where $B = 200$ following the configuration of the RWFNs. Therefore, the number of parameters in RWFNs is $(2n + 3) \cdot B = (2 \cdot 64 + 3) \cdot 200 = 26200$. Although our method requires more space complexity compared to LTNs ($26200 > 24972$), the parameters $\{\mathbf{W}, \mathbf{R}, \mathbf{b}\}$ in RWFNs are randomly drawn and fixed weights. Thus, it is also necessary to compare the number of learnable parameters across the two models.

All of the above parameters of LTNs must be adaptable, whereas the parameters to learn in RWFNs for object type classification are only $\boldsymbol{\beta} \in \mathbb{R}^{2B}$. Thus, the number of learnable parameters is 400, which is much smaller than that of LTNs. This means that the ratio of the two numbers of parameters to learn is about $400 : 24972 \approx 1 : 62$. Consequently, non-adaptable parameters in RWFNs can have significant power to represent the latent relationship among objects so that the model can efficiently extract relational knowledge even though using fewer adaptable

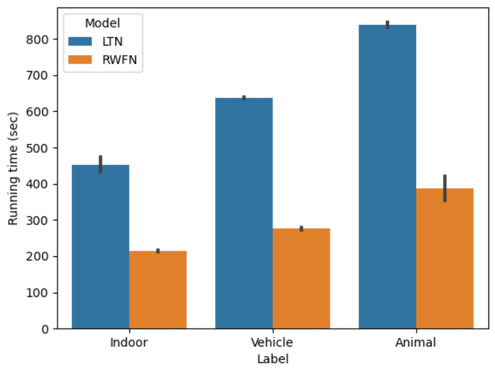


Figure 3: The comparison of running time, including data configuration time, training time (sec) for LTN and RWFN

parameters. Furthermore, the number of LTN parameters heavily depends on the number of features, whereas RWFNs 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.

Running Time Fig. 3 depicts the comparison of running time, including data configuration time and training time for LTNs and RWFNs. The running time of RWFNs is roughly as half of that of LTNs. This is because the number of learnable parameters in RWFNs is far smaller than LTNs, and RWFNs have linear models to learn, which is much simpler compared to models used in LTNs.

Space Complexity of RWFNs with Weight Sharing Weight sharing is a unique feature of RWFNs, which can greatly reduce necessary space complexity when multiple classifiers are used simultaneously. In the depicted case of learning i classifiers in Fig. 1b, the space complexity for RWFNs without using weight sharing is $(2 \cdot n \cdot B + 3 \cdot B) \cdot i = (2 \cdot n + 3) \cdot B \cdot i \approx O(i \cdot B \cdot n)$. However, with weight sharing, RWFNs can achieve much better space complexity, which is $2 \cdot n \cdot B + B + 2 \cdot B \cdot i \approx O(B \cdot n)$ because $B \cdot i < B \cdot n$ for the experiments conducted in the SII task, but it can also be $O(i \cdot B)$ for the different task. This indicates that the one of the factors that highly influence to the space complexity of the original RWFNs can be negligible when using weight sharing, which makes the model cost efficient and economical.