Report of Verifying the Results of *Embedding Symbolic Knowledge into Deep Networks*

The paper *Embedding Symbolic Knowledge into Deep Networks* proposes to leverage prior symbolic knowledge to improve the performance of deep models through embedding technology. In specific, it embeds propositional logic formula, which is converted to a rooted directed acyclic graph (DAG) at first, by using a multi-layer Graph Convolutional Network (GCN). What's more, two modifications, Heterogeneous Node Embedder and Semantic Regularization, are proposed for Deterministic Decomposable Negation Normal Form (d-DNNF) formula which is tractable for most queries. Two experiment are conducted to verify their effectiveness.

1. Synthetic Dataset

The goal of this experiment is to validate that d-DNNF formulae embeddings are more informative relative to embeddings of general form and CNF formulae. The authors trained the model to predict whether an assignment τ satisfies a formula f when given the embedding of f and the embedding of τ. The model is a 2-layer neural network with 150 hidden units. The model has different performance when the embeddings come from different embedders, including 7 different models using general, CNF, and d-DNNF formulae (with and without Heterogenous Node embedding and Semantic Regularization). Datasets are also divided into three different datasets ("low", "moderate", and "high") based on complexity. The complexity is reflected by the formula's number of variables n_v and the maximum depth d_m, $(n_v, d_m) = (3, 3), (3, 6), (6, 6)$.

Table 1 is the one in paper, and Table 2 summarizes the results of the experiments conducted by myself. First of all, I think Heterogeneous Node is also applicable for General formulae, so I tested it too. Second, when I read the code provided by authors, I found the definition of semantic regularizer loss is not consistent with that in paper, thus, I tested both of them. It seems that they do not make much effect on the accuracy. By the way, $q(\tau_F)$ and $q(\tau_T)$ seem to have to swap their positions in equation (3) introduced in paper, named by triplet loss, if we want to minimize distances between the embeddings of formulae and satisfying assignments.

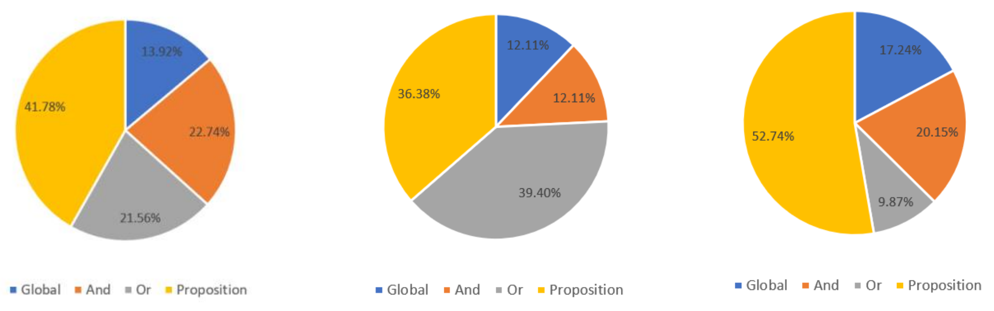
Table 1: Prediction accuracy and standard error over 10 independent runs with model using different forms of formulae and regularization. Standard error shown in brackets. "HE" means the model is a heterogeneous embedder, "SR" means the model is trained with semantic regularization. "✓" denotes "with the respective property" and "-" denotes "Not Applicable". The best scores are in bold.

Formula Form	HE	SR	Acc. (%)		
			Low	Moderate	High
General	-	-	89.63 (0.25)	70.32 (0.89)	68.51 (0.53)
CNF		-	90.02 (0.18)	71.19 (0.93)	69.42 (1.03)
	✓	-	90.25 (0.15)	73.92 (1.01)	68.79 (0.69)
d-DNNF			89.91 (0.31)	82.49 (1.11)	70.56 (1.16)
		✓	90.22 (0.23)	82.28 (1.40)	71.46 (1.17)
	✓		90.27 (0.55)	81.30 (1.29)	70.54 (0.62)
	✓	✓	**90.35 (0.32)**	**83.04 (1.58)**	**71.52 (0.54)**

Table 2: Prediction accuracy and standard error over 10 independent runs of my experiments. All symbols and abbreviations are the same as Table 1. “*” denotes “another definition of semantic regularizer loss”.

Formula Form	HE	SR	Acc. (%)		
			Low	Moderate	High
General		-	84.76 (0.93)	68.51 (2.05)	73.87 (1.81)
	✓	-	85.32 (0.70)	67.65 (1.77)	74.11 (1.21)
CNF		-	85.95 (1.89)	71.76 (2.82)	74.44 (2.10)
	✓	-	86.55 (0.77)	75.74 (3.58)	75.19 (1.38)
d-DNNF			92.63 (4.15)	90.19 (2.01)	79.25 (1.70)
		✓	**93.14 (3.95)**	**91.65 (1.18)**	79.69 (1.12)
	✓		86.69 (1.08)	91.63 (1.44)	**80.77 (1.29)**
	✓	✓	87.35 (1.01)	90.55 (1.61)	78.14 (0.88)
d-DNNF*		✓	92.21 (4.43)	90.44 (0.98)	79.43 (1.47)
	✓	✓	86.43 (1.34)	91.33 (1.87)	79.33 (1.47)

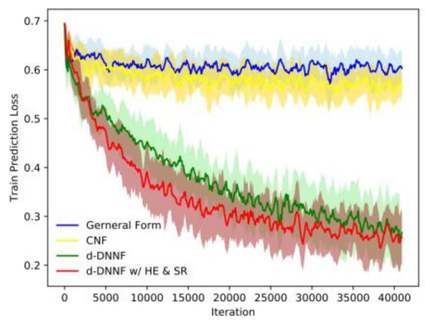
First, there is no doubt that the embedder trained on d-DNNF formulae performs better than other formulae, and Heterogeneous Node and Semantic Regularization improve embedder’s accuracy respectively. However, it is weird that the heterogeneous embedder with Semantic Regularization does not achieve highest accuracy. I have been trying to figure out the cause of this problem, but I have not succeeded so far. Besides, it can be noted that in “low” dataset, Heterogeneous Node embedding decreases the accuracy of d-DNNF formulae embedder. Small share of $\vee$ in the formulae may be the cause of this problem (Fig. 1), because the optimizer might cannot find a set of suitable parameters for the $\vee$ without enough samples.



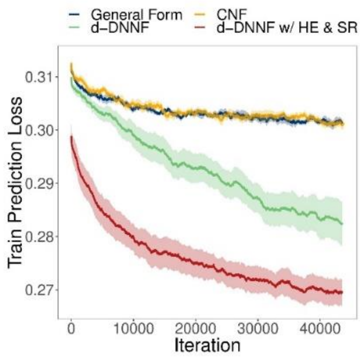
(a) (b) (c)

Figure 1: Average share of different types of nodes in logic graph in “Low” dataset. (a) General form; (b) CNF form; (c) d-DNNF form.

Fig. 2a illustrates that the d-DNNF embedder, especially with both Semantic Regularization and Heterogeneous Node, converges faster than the CNF and general form. It is consistent with the description in paper. Fig. 3a and Fig. 3b, which show the resultant embedding spaces are more informative of satisfiability, are also very similar, except that the range of Y-axis in Fig. 3a is from 0 to 1 while the other is from 0.5 to 1. From my point of view, it is possible and reasonable that there are some assignments that do not satisfy any of the clauses, i.e. percentage of satisfied clauses equals 0. The accuracies of d-DNNF embeddings are indeed higher that others, and the CNF and general form model is sensitive to the depth (Fig. 4). The depth changes from 3 to 6, i.e. from “low” dataset to “moderate” dataset, decreases the accuracy of them quite a lot; I think that is because the number of nodes in their logic graphs increases very much while that of d-DNNF does not (Fig. 5). Maybe it is a perspective to understand the *Embeddable-Demanding* concept proposed by authors.

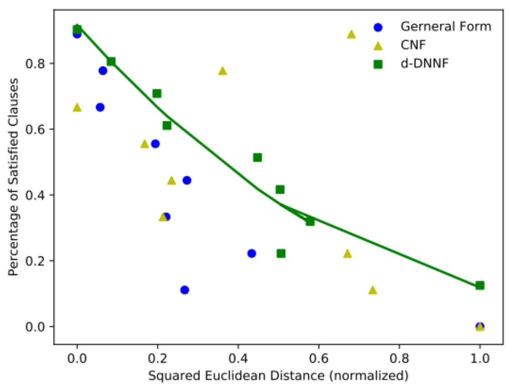


(a)

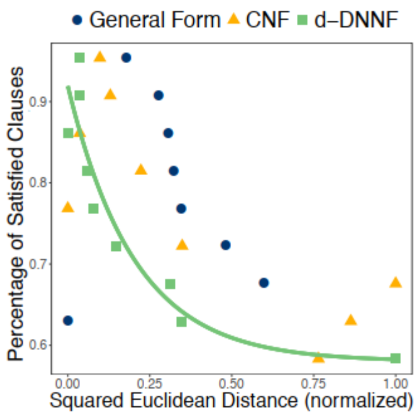


(b)

Figure 2: (a) Prediction loss as training progressed (line shown is the average over 10 runs with shaded region representing the standard error) in my experiments; (b) The corresponding one in paper.

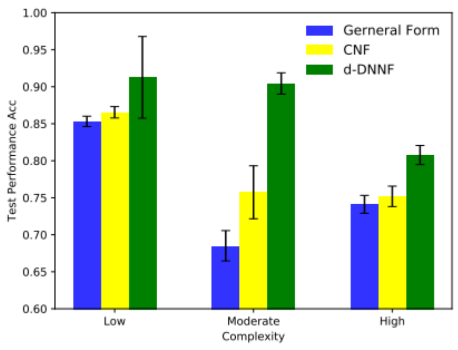


(a)

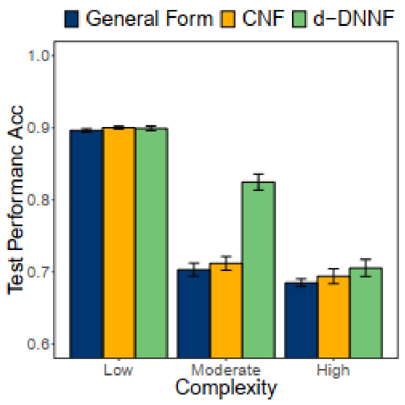


(b)

Figure 3: Formulae satisfiability v.s. distance in the embedding space, showing that LENSR learnt a good representation by projecting d-DNNF logic graphs. (a) The results in my experiment; (b) The corresponding one in paper.



(a)



(b)

Figure 4: (a) Test accuracies in my experiment; (b) The corresponding one in paper.

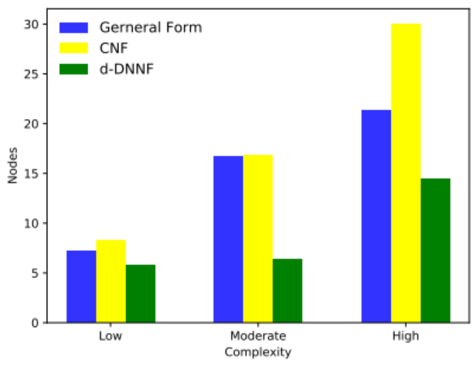


Figure 5: The number of nodes in logic graphs.

I wonder why the semantic regularizer loss only concerns d-DNNF formulae and ignore the assignments which are a conjunction of propositions. At the beginning, I carelessly calculated the semantic regularizer loss for both d-DNNF formulae and the assignments. As a result, the accuracy of models which are trained with semantic regularizer loss decrease by about 5%. It took me several weeks to realize this wrong calculation.

Inspired by Semantic Regularization, I am thinking whether it is possible to utilize direction attribute of the logic graph in the embedding structure since GCN can only process undirected graph. Another idea is that, for formulae which have n propositions, what will happen if we firstly fix the 2^n assignments at specified points which are evenly distributed throughout embedding space. Whether we can get a better embedding? If it works, whether we can further utilize this embedding to find satisfying and unsatisfying assignments for a formula, for example checking the closed and furthest assignments point.

2. Visual Relation Prediction

The task of Visual Relation Prediction (VRP) is to predict the visual relation between two objects given visual information in an input image. It is used to show how LENSR can be applied to a real-world task. The model is trained with both training data and prior knowledge which is in the form of logic formulae. The dataset named VRD contains 5,000 images, 4,000 for training and 1,000 for testing. However, limited by performance and capacity of my laptop, I can only run the code on 10% of the training set. Thus, my experiment results (Table 3) are based on 10% training set, but on 100% testing set. Table 4 is the corresponding one in paper. Except for top-5 accuracy score, I also tested the top-1 accuracy score to better compare the different models.

Although LENSR with Heterogeneous Node embedding and Semantic Regularization achieves the highest score, the gap between different models is very small, and the scores are also higher than those in the paper. One possible reason is that the reduction of the training set reduces the clauses and variables, and then makes the model simple and efficient, preventing over-fitting. Another weird phenomenon is that the test accuracy of embedders reaches 100% after only one training epoch. Is it because the size of training set is too small?

Table 3: Performance of VRP under different configurations in my experiments. “HE” indicates a Heterogeneous Node embedder, “SR” means the model uses semantic regularization. “X” denotes “with the respective property” and “-” denotes “Not Applicable”. The best scores are in bold.

Model	Form	HE	SR	Top-5 Acc. (%)	Top-1 Acc. (%)
without logic	-	-	-	96.60	86.11
with semantic loss	-	-	-	96.63	83.09
with treeLSTM embedder	CNF	-	-	95.81	85.22
	d-DNNF	-	-	95.87	86.66
LENSR	CNF		-	96.54	82.15
		✓	-	96.68	86.15
	d-DNNF			96.74	**86.81**
			✓	96.72	85.99
		✓		96.72	85.56
		✓	✓	**96.77**	85.43

Table 4: Performance of VRP under different configurations in the paper. All symbols and abbreviations are the same as Table 3.

Model	Form	HE	SR	Top-5 Acc. (%)
without logic	-	-	-	84.30
with semantic loss	-	-	-	84.76
with treeLSTM embedder	CNF	-	-	85.76
	d-DNNF	-	-	82.99
LENSR	CNF		-	85.39
		✓	-	85.70
	d-DNNF			85.37
			✓	88.01
		✓		90.13
		✓	✓	**92.77**