

1 Supplementary Material

2 1.1 Embedder Training Algorithm

3 The algorithm for training the embedder is summarized in Algorithm 1.

Algorithm 1 trainEmbedder

Input: f : CNF formula; I : Training images; m : Margin

Output: q : Embedder

```

1:  $q \leftarrow \text{init}()$ 
2: repeat
3:   for all  $I_i \in I$  do
4:     // Create intermediate formula
5:      $f_i \leftarrow \{ \}$ 
6:      $objs \leftarrow \text{all\_instances\_in}(I_i)$ 
7:     for all  $c_i \in f$  do
8:       if  $\text{all\_instances\_in}(c_i) \in objs$  then
9:          $f_i \leftarrow f_i \cup c_i$ 
10:      end if
11:    end for
12:     $f_i \leftarrow \text{append\_constraints}(f_i)$ 
13:     $\tau_T \leftarrow \text{sat\_assig\_of}(f_i)$ 
14:     $\tau_F \leftarrow \text{unsat\_assig\_of}(f_i)$ 
15:     $\Theta_q \leftarrow \underset{\Theta_q}{\text{argmin}} L_{emb}$ 
16:     $q \leftarrow \text{update\_with}(q, \Theta_q)$ 
17:  end for
18: until Convergence
19: return  $q$ 

```

4 1.2 Embeddable-Demanding

5 The significant performance improved due to usage of d-DNNF raises the question whether the
6 language represented by d-DNNF has a smaller search space and therefore, potentially easier learning
7 method. To this end, we introduce the concept of embeddable-demanding below

8 The following theorems uses the standard complexity theoretic terms and we refer to the reader to the
9 standard text [11] for detailed treatment of these concepts.

10 **Definition 1 (Embeddable-Demanding)** Let L_1, L_2 be two compilation languages. L_1 is at least
11 as embeddable-demanding as L_2 iff there exists a polynomial p such that for every sentence $\alpha \in$
12 $L_2, \exists \beta \in L_1$ such that (i) $|\beta| \leq p(|\alpha|)$. Here $|\alpha|, |\beta|$ are the sizes of α, β respectively, and β may
13 include auxiliary variables. (ii) The transformation from α to β is poly time. (iii) There exists a
14 bijection between models of β and models of α .

15 **Theorem 1.1** CNF is at least as embeddable-demanding as d-DNNF but if d-DNNF is at least as
16 embeddable-demanding as CNF then $P = PP$

17 **Proof 1.1** (1) Prove that CNF is at least as embeddable-demanding as d-DNNF, i.e. for every formula
18 α in d-DNNF, there exists a polynomial size, and polynomial time computable CNF formula β such
19 that there is an one to one polynomial time computable mapping between models of β to α .

20 Observe that d-DNNF represents a circuit, which can be encoded into an equisatisfiable CNF formula
21 of polynomial size due to NP-completeness of CNF. In particular, the usage of Tseytin encoding [2]
22 ensures that the resulting CNF is of linear size. Furthermore, let d-DNNF G be defined over the set
23 of variables denoted by X , then Tseytin encoding introduces a set of auxiliary variables, say Y , for
24 the resulting formula F such that $G(X) = \exists Y F(X \cup Y)$. Therefore, the mapping from models of G
25 to F is achieved just by projection of models of G on X .

26 (2) Prove that if d -DNNF is at least as embeddable-demanding as CNF then $P = PP$. In other
 27 words, if for every formula β in CNF, there exists a polynomial size, and polynomial time computable
 28 d -DNNF α such that there is bijection between models of α and models of β , then $P = PP$. $P = PP$
 29 implies collapse of entire polynomial hierarchy, in particular $P = NP$.

30 Assume for every formula β in CNF, there exists a polynomial size, and polynomial time computable
 31 d -DNNF α such that there is a bijection between models of α and models of β . Since d -DNNF allows
 32 counting in polynomial time and the existence of bijection implies that the number of models of α
 33 is equal to that of β , then we can compute the number of models of an arbitrary CNF formula in
 34 polynomial time; therefore $P = PP$. In this context, it is worth noting that the entire polynomial
 35 polynomial hierarchy is shown to contain PP , i.e., $PH \subseteq PP$ [3].

36 1.3 Computing Infrastructure

37 We trained our models using Pytorch 0.4.1 on one NVIDIA GTX 1080 Ti 12GB GPU.

38 1.4 Hyper-parameters Selection

39 Our hyper-parameters includes: the margin in triplet loss of the embedder m , the semantic regularizer
 40 weight λ_r and logic loss weight λ . The ranges considered are $[0.5, 5]$ for m ; $[0.05, 0.2]$ for λ_r and
 41 $[0.05, 0.2]$ for λ . We did grid search and set $m = 1.0$, $\lambda_r = 0.1$, $\lambda = 0.1$ across all experiments.

42 References

- 43 [1] S. Arora and B. Barak, *Computational complexity: a modern approach*. Cambridge University
 44 Press, 2009.
- 45 [2] G. S. Tseytin, *On the Complexity of Derivation in Propositional Calculus*, pp. 466–483. Berlin,
 46 Heidelberg: Springer Berlin Heidelberg, 1983.
- 47 [3] S. Toda, “Pp is as hard as the polynomial-time hierarchy,” *SIAM Journal on Computing*, vol. 20,
 48 no. 5, pp. 865–877, 1991.