Deepire II = RL(GNN+2RvNN)[*]

Martin Suda

Czech Technical University in Prague, Czech Republic

Deepire II is a new neural architecture implemented in the saturation-based theorem prover Vampire [2] for clause selection guidance [3]. It consists of a Graph Neural Network (GNN) for a name-invariant processing of the input Clause Normal Form (CNF), followed by two Recursive Neural Networks (RvNN): one running along clause derivation history (as in Deepire I [12, 13]) and one along the clause term structure (as in ENIGMA-NG [5]). Trained on the standard TPTP benchmark for first-order proving, in a setup inspired by the Reinforcement Learning (RL) paradigm, Deepire II proves 20% more theorems than the baseline strategy.

The RL-inspired learning operator comes from the previous AITP [14]. Also the current architecture was already sketched there, however, an efficient implementation took longer to accomplish and the results reported here are from a submission to CADE 2025 [15].

One-off GNN Invocation. Graph Neural Networks (GNNs) have been established as good basis for name invariant neural formula representations [10, 7, 1]. However, GNNs are relatively expensive to evaluate and intuitively work best when given a large formula (i.e., many clauses) to evaluate at once, so that individual constituents (sub-formulas, terms, symbols) provide sufficient context for one another [4, 6]. This becomes tricky when evaluating clauses for clause selection, as the group of newly derived clauses, which need to be evaluated after each activation, varies in size and is typically relatively small.

While still taking advantage of GNNs' representational power, Deepire II avoids these complications by running a GNN only once, at the beginning of the saturation process on the input problem's CNF, and have the GNN prepare name-invariant *embeddings* of the signature symbols and the input clauses, to be further utilized in subsequent processing.

Generalizing Age and Weight with RvNNs. The idea is to use these embeddings to seed computations of two independent Recursive Neural Networks (RvNN) [9].

As mentioned, one RvNN starts from the input clause embeddings and unrolls along the clause derivation tree, the other starts from the symbol embeddings and unrolls along the clause parse tree. In fact, it is advantageous to treat each of these trees as a directed acyclic graph (DAG), share the common subgraphs and cache the results for the already computed subgraphs.

One can think of these two RvNNs as allowing the training process to generalize and improve upon the two most commonly used standard Clause Evaluation Functions (CEFs), clause *age* and clause *weight*. The former is essentially the depth of the derivation tree, while the latter equals the number of nodes in the parse tree. As such, each could in principle be rediscovered by the new architecture through learning, which intuitively feels like a good design.

Completing the Neural CEF. To complete the tower of NN modules and obtain a final score for a clause, Deepire II concatenates the embeddings from the two RvNNs with a vector of 12 easy-to-compute simple clause features. These features include the standard age and weight, the number of literals based on their polarity, the number of variable occurrences or the number of AVATAR [17] splits a clause depends on. The concatenated vector is then fed to a simple fully-connected NN (an MLP) with one hidden layer and a single clause score output.

[*]This work was supported by the Czech Science Foundation project no. 24-12759S.

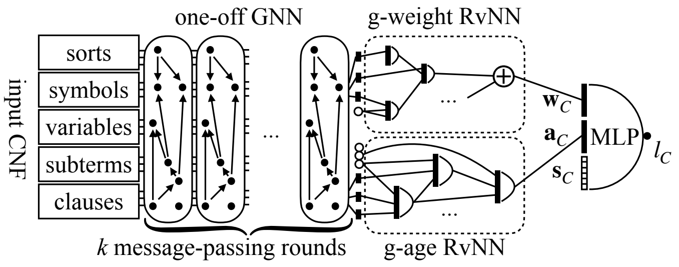

Figure 1: Clause evaluation neural architecture. Information flows from left to right.

The whole architecture is shown in Figure 1. In the experiments, the number of GNN message-passing rounds was set to $k = 8$, embeddings had size $n = 32$ (both in the GNN and in the RvNNs), and non-linearities were computed using ReLU after a temporary linear expansion to size $m = 256$. Please consult the CADE submission for additional details [15]. The architecture was implemented in PyTorch 2.5 library [11] and we used the TorchScript bridge for interfacing the neural model trained in Python from C++, in which Vampire is written.

Experimental Results. We used Vampire's default strategy under an instruction limit of 30 000 Mi ($\sim$ 10 s) per proof attempt both as the source of initial training traces, as well as the basis for the neurally guided extension (so that the extra time spent 'thinking' has to be made up by better decisions). We took all the 19 477 first-order problems from TPTP library [16] v9.0.0 and randomly split them into 15 000 training problems and the rest for testing.

The default strategy solved approximately 46.3 % of the training problems, while Deepire II, after 24 improvement iterations (i.e., iteratively training from traces obtained by the previous version of the guided prover and evaluating to collect more traces), solved 56.2 % of the training and 54.5 % of the testing problems. The best test-performance gain (of the advertised 20 % over the baseline) was obtained by implementing two additional tricks: We used several repeated randomized runs of the prover to collect more training traces in each iteration and we assigned more weight in the training to traces from problems solved only in recent loop iterations, while the weight of problems solved repeatedly in past successive loops was gradually decreased.

Solving Hard Problems. To check whether our neural guidance can also help Vampire solve hard problems, we ran one more training session, but this time with runs limited at 100 000 Mi ($\sim$ 40 s) on the whole first-order TPTP.[1] During the 30 improvement loops for which this experiment ran, Deepire II solved 130 TPTP problems of rating 1.0.

A TPTP problem having rating 1.0 means that it has not been solved during the last rating evaluation [16]. While there are many rating 1.0 problems that had been solved at some point in the past, out of our 130 problems, we found 49 that were actually never solved even once! This is a remarkable occurrence in the context of a single strategy improvement.

Outlook. Our new architecture involves many design decisions and hyperparameters that can be questioned and tuned. While we value the unprecedented success on the diverse—and usually hard to learn for—TPTP benchmark, evaluating Deepire II on some of the more common sets problems in this context, such as Mizar40 [8], is the obvious next step.

[1]Note that not using a train/test split is fine here, because none of the problems we now report were solved by the seeding default strategy and so they had to be attained solely thanks to generalization.

References

[1] I. Abdelaziz, M. Crouse, B. Makni, V. Austel, C. Cornelio, S. Ikbal, P. Kapanipathi, N. Makondo, K. Srinivas, M. Witbrock, and A. Fokoue. Learning to guide a saturation-based theorem prover. *IEEE Trans. Pattern Anal. Mach. Intell.*, 45(1):738–751, 2023.

[2] F. Bartek, A. Bhayat, R. Coutelier, M. Hajdu, M. Hetzenberger, P. Hozzova, L. Kovacs, J. Rath, M. Rawson, G. Reger, M. Suda, J. Schoisswohl, and A. Voronkov. The vampire diary. In *CAV 2025*, 2025. To appear.

[3] L. Blaauwbroek, D. M. Cerna, T. Gauthier, J. Jakubův, C. Kaliszyk, M. Suda, and J. Urban. Learning guided automated reasoning: A brief survey. In *Logics and Type Systems in Theory and Practice - Essays Dedicated to Herman Geuvers on The Occasion of His 60th Birthday*, vol. 14560 of *LNCS*, pp. 54–83. Springer, 2024.

[4] K. Chvalovský, J. Jakubův, M. Olšák, and J. Urban. Learning theorem proving components. In *TABLEAUX 2021*, vol. 12842 of *LNCS*, pp. 266–278. Springer, 2021.

[5] K. Chvalovský, J. Jakubův, M. Suda, and J. Urban. ENIGMA-NG: efficient neural and gradient-boosted inference guidance for E. In *CADE 2019*, vol. 11716 of *LNCS*, pp. 197–215. Springer, 2019.

[6] A. Fokoue, I. Abdelaziz, M. Crouse, S. Ikbal, A. Kishimoto, G. Lima, N. Makondo, and R. Marinescu. An ensemble approach for automated theorem proving based on efficient name invariant graph neural representations. In *IJCAI 2023*, pp. 3221–3229. ijcai.org, 2023.

[7] J. Jakubův, K. Chvalovský, M. Olšák, B. Piotrowski, M. Suda, and J. Urban. ENIGMA Anonymous: Symbol-independent inference guiding machine. In *IJCAR 2020*, vol. 12167 of *LNCS*, pp. 448–463. Springer, 2020.

[8] C. Kaliszyk and J. Urban. Mizar 40 for mizar 40. *J. Autom. Reason.*, 55(3):245–256, 2015.

[9] A. Küchler and C. Goller. Inductive learning in symbolic domains using structure-driven recurrent neural networks. In *KI 1996*, vol. 1137 of *LNCS*, pp. 183–197. Springer, 1996.

[10] M. Olšák, C. Kaliszyk, and J. Urban. Property invariant embedding for automated reasoning. In *ECAI 2020*, vol. 325 of *Frontiers in Artificial Intelligence and Applications*, pp. 1395–1402. IOS Press, 2020.

[11] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

[12] M. Suda. Improving ENIGMA-style clause selection while learning from history. In *CADE 2021*, vol. 12699 of *LNCS*, pp. 543–561. Springer, 2021.

[13] M. Suda. Vampire with a brain is a good ITP hammer. In *FroCoS 2021*, vol. 12941 of *LNCS*, pp. 192–209. Springer, 2021.

[14] M. Suda. Two learning operators for clause selection guidance: An experimental evaluation. In *9th Conference on Artificial Intelligence and Theorem Proving AITP 2024 – proceedings*, 2024. https://aitp-conference.org/2024/abstract/AITP_2024_paper_4.pdf.

[15] M. Suda. Efficient neural clause-selection reinforcement. In *Proceedings of the 30th International Conference on Automated Deduction (CADE-30)*, 2025. To appear. Preprint: https://arxiv.org/abs/2503.07792.

[16] G. Sutcliffe. Stepping Stones in the TPTP World. In *IJCAR 2024*, number 14739 in LNAI, pp. 30–50, 2024.

[17] A. Voronkov. AVATAR: the architecture for first-order theorem provers. In *CAV 2014*, vol. 8559 of *LNCS*, pp. 696–710. Springer, 2014.