Two Learning Operators for Clause Selection Guidance: An Experimental Evaluation*

Martin Suda

Czech Technical University in Prague, Czech Republic

Clause selection is arguably the most important choice point in saturation-based automated theorem proving [9]. It amounts to deciding, in each iteration of the saturation loop, which next clause to *select* for *activation*, i.e., for the promotion from the *passive* set to the *active* set, which means enabling this selected clause's participation in generating inferences. In the commonly used Discount loop [1], a refutation can only be successfully completed when all clauses that the proof consists of have been selected. This means that in order to get a more powerful prover, "all we need to do" is to come up with a heuristic which is better than our current ones at selecting clauses that will most likely end up in the yet to be discovered proof.

Recently, several successful systems for machine-learned clause selection guidance have been developed, most notably ENIGMA in several incarnations [5, 6, 7, 4] and Deepire [10, 11]. Although the underlying machine learning models differ (e.g., boosted trees, neural network, etc.), the systems all learn from previously observed prover derivations, training a *binary classifier* to recognize as positive those clauses that appeared in the discovered proofs, against the background of all the recorded selected clauses. Although this is typically not stated explicitly, the learning setup in ENIGMA or Deepire can be seen to already assume a working clause selection heuristic and mainly seeks to improve upon it through the integration of the learned advice.

On the other hand, approaches inspired by the reinforcement learning (RL) paradigm [14, 15], strive to learn a standalone clause selection heuristic from scratch [3, 12]. Although the found proof is still used as the gold positive label for the clauses which appear in it, the background against which the model learns consists of the passive clauses (as opposed to the selected and activated ones) and, moreover, each recorded derivation step can use the precise snapshot of the passive set content at the given moment. This RL-inspired approach a priory leads to a *regression* model (as opposed to a classifier), although the technical details (notably the loss function) are surprisingly similar.

Despite the mentioned differences, both the ENIGMA-style and the RL-inspired learning operators ultimately aim to achieve the same pragmatic goal, namely to improve the prover performance by learning from experiences gathered while solving problems from some benchmark family or distribution of interest. However, to the best of my knowledge there is no direct experimental comparison of the two. During my talk at WAPSML this March in Vienna[1] I managed to compare and contrast ENIGMA-style and the RL-inspired learning from the design perspective, but the performance evaluation remained unexplored. I would like to pay off this debt to the scientific audience at AITP 2024.

There are at least two dimensions to such an evaluation. It seems that the RL-inspired approach is more faithful in positively boosting the decisions that would lead to the discovered proof in the precise contexts given by the evolving content of the passive set, while the ENIGMA-style one is based on a coarser proxy. If this intuitions holds, the main question would be whether the extra precision of the RL-inspired approach in the end actually pays off performance-wise, and secondarily, whether the overhead connected with its more complicated setup does not render the training procedure too slow or memory inefficient to be of practical value.

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

[1] https://europroofnet.github.io/wg5-vienna24/

(Recall that a modern prover may easily generate hundred thousand clauses within few seconds, which mostly occupy the passive set and compete for being selected.)

Deepire 2.0

In the past, I presented at AITP my system Deepire [10, 11], an instance of ENIGMA-style learning for the ATP Vampire [8], which distinguishes itself from other work by building recursive neural networks (RvNN) along clause derivation history as the basis for clause classification. More recently, I examined the RL-inspired approach, also for Vampire, by using a few simple-to-compute numerical clause features that a small MLP then turns into clause logits for sorting the passive set [12, 13]. For a fair comparison of the two learning approaches, I will bring the essential components from the two ML-guided Vampires into one place.

In a first stage, since ENIGMA-style learning is easier to manage than the RL-inspired approach and, at the same time, MLPs are conceptually simpler than RvNNs, I will focus on porting the various ways of integrating the learned advice explored in the Deepire work [10] to also function with the simple clause features and to co-exists with the RL-inspired codebase. To recall, with ENIGMA-style learning one has the possibility to either fully trust the newly learned heuristic (treating the binary classifier's outputs as a clause score for sorting a new queue) or to combine it with the old one in some alternating fashion. Ultimately, I expect the layered approach with lazy evaluation to be the best performing mode of integration [10]. Since the simple clause features have already been shown to work in RL-inspired approach, adding them into an ENIGMA-style setting will be enough to get a first fair comparison.

For a second, more challenging, stage, I will go in the other direction and make my RL-inspired approach implementation compatible with a more complex way of evaluating clauses. I intend to take this as an opportunity to go beyond any neural architecture for clause selection guidance considered in the past. Keeping efficiency of evaluation in mind and also respecting the now well-understood requirement of name invariance [4], I am proposing an architecture to consists of (1) a GCN to embed the input problem's formulas as well as its signature symbols in a pre-saturation one-off invocation, followed by (2) two RvNNs, one evolving along the term and clause structure of every generated clause (as in [2]) and the other along the clause derivation history (as in Deepire). Both the RvNNs will take the respective name-invariant embeddings from the GCN (1) as input (symbol embeddings to seed the term derivation and formula embeddings to seed the clause derivation history). Note that the idea is to run the relatively expensive GCN only once at the beginning (and only on the input formulas and not on every clause derived during saturation) to save time. Maintaining the locally evolving RvNNs then only requires an amount of computation proportional in size to the data processed by the ATP anyway, and, moreover, is well suited for speedups through the use of caching [2, 10].

While this new architecture is meant to be relatively efficient to evaluate, the real test will come with training it end-to-end, i.e., connecting the RvNNs (spanning possibly millions of terms and thousands of clauses) with the GCN (spanning several iterations of message passing over the input formulas construed as a graph) into one differentiable computation. If that turns out to be feasible, and not just for the ENIGMA-style but also for the RL-inspired case, where we also need to track the evolving state of the passive set, this will provide for a very realistic setup for comparing these two approaches.

As a final note, let us recall that while the RL-inspired approach is usually intended as "ever improving" in the sense that a new version of the guiding agent is used to prepare more training data for further reinforcement, in the ENIGMA-style learning, we achieve the same effect with the technique named "looping" [7]. A fair comparison only comes from running each of the operators until they reach their maximal performance and ideally stabilize there.

References

[1] J. Avenhaus, J. Denzinger, and M. Fuchs. DISCOUNT: A system for distributed equational deduction. In *Rewriting Techniques and Applications, 6th International Conference, RTA-95, Kaiserslautern, Germany, April 5-7, 1995, Proceedings*, vol. 914 of *Lecture Notes in Computer Science*, pp. 397–402. Springer, 1995.

[2] K. Chvalovský, J. Jakubuv, M. Suda, and J. Urban. ENIGMA-NG: efficient neural and gradient-boosted inference guidance for E. In *Automated Deduction - CADE 27 - 27th International Conference on Automated Deduction, Natal, Brazil, August 27-30, 2019, Proceedings*, vol. 11716 of *Lecture Notes in Computer Science*, pp. 197–215. Springer, 2019.

[3] M. Crouse, I. Abdelaziz, B. Makni, S. Whitehead, C. Cornelio, P. Kapanipathi, K. Srinivas, V. Thost, M. Witbrock, and A. Fokoue. A deep reinforcement learning approach to first-order logic theorem proving. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pp. 6279–6287. AAAI Press, 2021.

[4] J. Jakubuv, K. Chvalovský, M. Olsák, B. Piotrowski, M. Suda, and J. Urban. ENIGMA anonymous: Symbol-independent inference guiding machine (system description). In *Automated Reasoning - 10th International Joint Conference, IJCAR 2020, Paris, France, July 1-4, 2020, Proceedings, Part II*, vol. 12167 of *Lecture Notes in Computer Science*, pp. 448–463. Springer, 2020.

[5] J. Jakubuv and J. Urban. ENIGMA: efficient learning-based inference guiding machine. In *Intelligent Computer Mathematics - 10th International Conference, CICM 2017, Edinburgh, UK, July 17-21, 2017, Proceedings*, vol. 10383 of *Lecture Notes in Computer Science*, pp. 292–302. Springer, 2017.

[6] J. Jakubuv and J. Urban. Enhancing ENIGMA given clause guidance. In *Intelligent Computer Mathematics - 11th International Conference, CICM 2018, Hagenberg, Austria, August 13-17, 2018, Proceedings*, vol. 11006 of *Lecture Notes in Computer Science*, pp. 118–124. Springer, 2018.

[7] J. Jakubuv and J. Urban. Hammering mizar by learning clause guidance (short paper). In *10th International Conference on Interactive Theorem Proving, ITP 2019, September 9-12, 2019, Portland, OR, USA*, vol. 141 of *LIPIcs*, pp. 34:1–34:8. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[8] L. Kovács and A. Voronkov. First-order theorem proving and vampire. In *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*, vol. 8044 of *Lecture Notes in Computer Science*, pp. 1–35. Springer, 2013.

[9] S. Schulz and M. Möhrmann. Performance of clause selection heuristics for saturation-based theorem proving. In *Automated Reasoning - 8th International Joint Conference, IJCAR 2016, Coimbra, Portugal, June 27 - July 2, 2016, Proceedings*, vol. 9706 of *Lecture Notes in Computer Science*, pp. 330–345. Springer, 2016.

[10] M. Suda. Improving enigma-style clause selection while learning from history. In *Automated Deduction - CADE 28 - 28th International Conference on Automated Deduction, Virtual Event, July 12-15, 2021, Proceedings*, vol. 12699 of *Lecture Notes in Computer Science*, pp. 543–561. Springer, 2021.

[11] M. Suda. Vampire with a brain is a good ITP hammer. In *Frontiers of Combining Systems - 13th International Symposium, FroCoS 2021, Birmingham, UK, September 8-10, 2021, Proceedings*, vol. 12941 of *Lecture Notes in Computer Science*, pp. 192–209. Springer, 2021.

[12] M. Suda. Elements of reinforcement learning in saturation-based theorem proving. In *7th Conference on Artificial Intelligence and Theorem Proving AITP 2022 – proceedings*, 2022. http://aitp-conference.org/2022/abstract/AITP_2022_paper_11.pdf.

[13] M. Suda. Descending to complementarity. In *8th Conference on Artificial Intelligence and Theorem Proving AITP 2023 – proceedings*, 2023. https://aitp-conference.org/2023/abstract/AITP_2023_paper_15.pdf.

[14] R. S. Sutton and A. G. Barto. *Reinforcement learning - an introduction*. Adaptive computation and machine learning. MIT Press, 1998.

[15] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8:229–256, 1992.