Fast weight programmers implementation

Evan M Drake

22/10/2023

Contents

1 Introduction

Meta-learning is a powerful tool in the presents of high training costs. Techniques like supervised-learning algorithms have training cost as a function of training data size. This is a consequence the sequential design pattern common in building these supervised-learning models [31], as seen below.

- Build a model
- Train the model
- Use the model

Consequently there are two ways to optimize this training cost problem: shorten the training phase, build a faster model. Meta-learning algorithms take advantage of the first method by utilizing few-shot learning [22], or getting rid of the training phase completely [39]. One such learning model is Schmidhuber's *Self referential weighted matrix*(SRWM), it does not require a training phase [1]. This machine can learn and adapt during experimentation. Here I will utilize an SRWM to play the game of American checkers[2]. The goal is to prove proposition 1 and all consequential propositions.

2 Objective

Proposition 1 *If a SRWM and a weighted matrix are to play the same game of checkers with no training phase the SRWM will play more efficiently.*

In proving proposition 1 we will have proven that for the game of American checkers a SRWM implemented as we do can play more efficiently than a standard weighted decision matrix. We will prove this by constructing both a weighted decision matrix and an SRWM and allowing them to play the same game of checkers. The outcome will be a proof by construction with the assumption that all outlier games will be ignored.
In particular we aim to set ground work for the proof that an SRWM can be used as a drop in replacment for any weighted decision matrix.

[1]SRWMs do not require a training phase to begin learning, but can still benefit from them

[2]American checkers or *English draughts* is explained in section 4

3 Literature Review

3.1 Adaptation Based Meta-learning

In adaptation based algorithms the underlying data structure (usually a network or set of networks) is never changed on a large scale but rather incrementally augmented until it fits the current task. Techniques using this approach tend to experience similar problems such as catastrophic memory loss[3], and change scale[4].

Schmidhuber [39] introduced the idea of a self modifying network that utilizes introspection to discover optimization probabilities. The idea is that given some weighted matrix $M(V,E)$[5], we can relate the change in weights for all E temporally. At a high level, this temporal analysis will expose a sort of gradient decent optimization problem [22]. The introspective phase is conducted with the instruction of special input units that outline a manner in which to analyze the network and output units that outline the modification of the matrix once analysis is complete. Input and output units are designed to describe the matrix by way of time-varying activations [6]. These activations serve to represent the connection weights and the weight modifications respectively for the special input and output units. **Irie et al.** [22] shows a more fine grained approach to Schmidhubers simple self modification, applying it to modern neural network training techniques and providing different implementation strategies. Irie et al. show the possibility that in certain situations self referential weighted matrices can not only adapt their learning methodologies but can also adapt the way they adapt their learning methodologies (meta-meta-learning). Their self-referential weighted matrix can be described as such:

$$y_t, k_t, q_t, \beta_t = \mathbf{W}_{t-1}\phi(x_t) \tag{1}$$
$$\hat{v}_t = \mathbf{W}_{t-1}\phi(k_t) \tag{2}$$
$$v_t = \mathbf{W}_{t-1}\phi(q_t) \tag{3}$$
$$\mathbf{W}_t = \mathbf{W}_{t-1} + \sigma(\beta_t)(v_t - \hat{v}_t) \otimes \phi(k_t) \tag{4}$$

[3]Memory loss is when the learning algorithm is allowed to modify to much information and a net information loss occurs.

[4]I define change scale as the problem of finding modification granularity

[5]Note that it is easier to think of M as a network

[6]Provided a constant stream of problems the paths taken can vary on each iteration

"where the $\otimes$ denotes the outer product, σ is the sigmoid function, and ϕ is an element-wise activation function whose output elements are all positive and sum up to one (e.g. softmax)" [22].

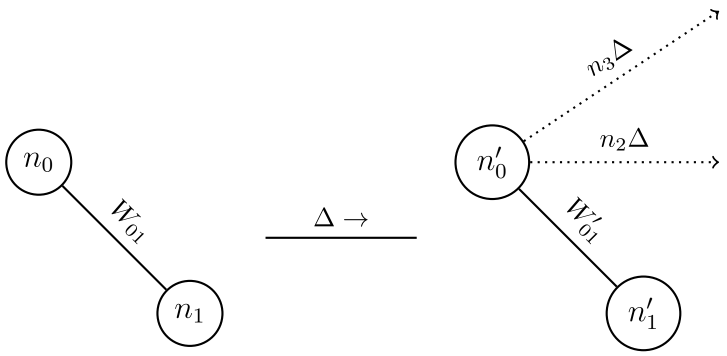


Figure 1: Δ describes how the graph will adapt over time. Each time step is associated with some problem in the constant problem stream, as the problem is solved the graph will adapt in a consistent way. The consistant way is some element (or combination there of) of the set of adaptation techniques, including but not limited too: { rewiring, attribute modification, node modification}. It is worth noting that degree preservation is not (usually) a goal of algorithms in this category.

4 Methods

4.1 American checkers rules

- played on an 8x8 American checkerboard
- 12 pieces per side
- pieces can move forward unless capturing
- to capture a piece one must cross another piece (of opposite loyalty) diagonally forward
- pieces that reach the opposite side promote to *kings*
- *kings* can move (as well as capture) forward or backward

4.2 Axiomatic strategy

I devise to construct a machine out of axiomatic modules, as a functional design. I have defined a board and the structure of the game, building several mapping functions that will progress the game, augment weights, and perform analysis. These mappings are non-bijective as any board of pieces can yield many temporally subsequent boards.

```haskell
  dem = nrows mb

-- | 'mapMat' is a map transform over any 'Matrix' a that
    ↪ exposes the row,column # to f
-- where the function exposes (row -> column -> 'Matrix' a -> b)
-- 'mapMat' will currently only take square matrices
mapMat :: Matrix a -> (Int -> Int -> Matrix a -> b) -> Matrix b
mapMat mb f = mapMatAux 0 0 mb f [] []

-- | 'move' moves a single specified piece on the board with a
-- move takes piece location, 'MoveType'
-- for captures right is column-1 and left is column+1 you can
    ↪ think of it as direction from whites
-- perspective.
```

Figure 2: Haskell map function for making adjustments over any preset checkerboard

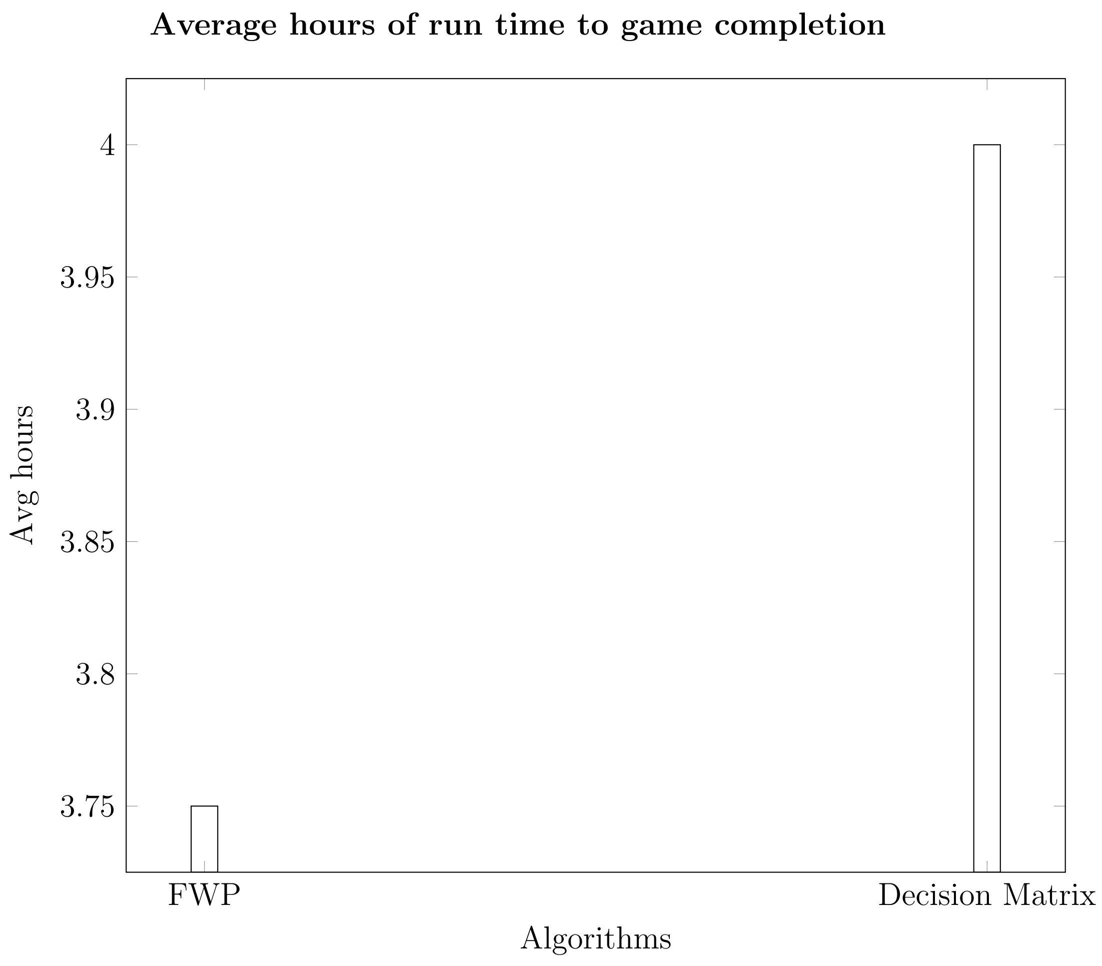
Utilizing mappings like the one in Figure 2 I can traverse the board in the same way for all algorithms. The mapping strategy also allows for easy-to-read code as functions can be built around the mapping. Weights can be adjusted with mappings however, the analysis step will require a reduction function with a similar signature.

4.3 Experiment parameters

The experiment is conducted as a comparison between two machines, a standard weighted decision matrix and an SRWM. The comparison is conducted on metrics pulled from games played. The machines will not store data between games but rather will run each game as a few-shot sequence. This way of conducting the experiment will allow for the most targeted testing for direct replacement of weighted decision matrices in standard trials.

5 Results

The tangible results from this experiment are negligible at best and misleading at worst. However for the purposes of this mock paper I will call them inconclusive. I was able to play the game twice with both version of the machine using GHCI to put the library together. The way I did this is not in the least reproducible, I used the interpreter and the structure of Haskell's monads system to (fake) a production ready system. Using this interpreter state I was able to play to completion two games with both machines, taking around three to four hours a game.



6 Conclusion

The overhead of integrating a Fast Weight Programmer is difficult to justify. The difficulty of integration, combined with the small improvement seen from my experimentation gives no evidence to the general superiority of FWPs over a decision matrix. I have observed that by the end of the testing games the FWP begins to give answers substantially fast and more precisely than the decision matrix, this indicates that given memory retention the FWP would continuously increase accuracy.

7 Future Work

This work is intended to lead to a detailed study into the likelihood of SRWMs being able to serve as a plugin replacement for weighted decision matrices. I do not intend to make any proof of this point however, my intention for this research is to prompt the question "are they better at other things too?". A good place for such research to start is to extend this experiment to a more complex game (perhaps Chess or Go). Another option is to expand proposition 1. The current experiment utilizes only few-shot learning. A possible expansion is to include learning phases and attempt the same test to see if we observe the same results.

I thought it was only right to include my whole meta-learning section here as I believe they all added inspiration

References

[1] *Alonzo Church ¿ D. The Λ-Calculus and Type Theory (Stanford Encyclopedia of Philosophy).* URL: `https://plato.stanford.edu/entries/church/supplementD.html#D2ChurSimpTheoType` (visited on 10/09/2023).

[2] Stanislaw Ambroszkiewicz. *The grounding for Continuum.* July 11, 2019. arXiv: `1510.02787[cs,math]`. URL: `http://arxiv.org/abs/1510.02787` (visited on 09/18/2023).

[3] D. M. Bossens, N. C. Townsend, and A. J. Sobey. "Learning to learn with active adaptive perception". In: *Neural Networks* 115 (2019), pp. 30–49. ISSN: 0893-6080. DOI: `https://doi.org/10.1016/j.neunet.2019.03.006`. URL: `https://www.sciencedirect.com/science/article/pii/S0893608019300796`.

[4] Philip K Chan and Salvatore J Stolfo. "Experiments on multistrategy learning by meta-learning". In: *Proceedings of the second international conference on {Information} and knowledge management - {CIKM} '93.* Washington, D.C., United States: ACM Press, 1993, pp. 314–323. ISBN: 978-0-89791-626-4. DOI: `10.1145/170088.170160`. URL: `http://portal.acm.org/citation.cfm?doid=170088.170160` (visited on 09/20/2023).

[5] Xiaohong Chen et al. "Towards a unified proof framework for automated fixpoint reasoning using matching logic". In: *Proceedings of the ACM on Programming Languages* 4 (OOPSLA Nov. 13, 2020), pp. 1–29. ISSN: 2475-1421. DOI: `10.1145/3428229`. URL: `https://dl.acm.org/doi/10.1145/3428229` (visited on 09/18/2023).

[6] Alonzo Church. *lambda-Calculus and Type Theory.* URL: `https://plato.stanford.edu/entries/church/supplementD.html` (visited on 10/09/2023).

[7] Kevin Clark et al. *Meta-Learning Fast Weight Language Models.* Dec. 5, 2022. arXiv: `2212.02475[cs]`. URL: `http://arxiv.org/abs/2212.02475` (visited on 09/24/2023).

[8] Michael T. Cox and Ashwin Ram. “Introspective multistrategy learning: On the construction of learning strategies”. In: *Artificial Intelligence* 112.1 (Aug. 1, 1999), pp. 1–55. ISSN: 0004-3702. DOI: 10.1016/S0004-3702(99)00047-8. URL: https://www.sciencedirect.com/science/article/pii/S0004370299000478 (visited on 09/20/2023).

[9] Haskell B. Curry. “Combinatory recursive objects of all finite types”. In: *Bulletin of the American Mathematical Society* 70.6 (1964), pp. 814–817. ISSN: 0273-0979, 1088-9485. DOI: 10.1090/S0002-9904-1964-11245-3. URL: https://www.ams.org/bull/1964-70-06/S0002-9904-1964-11245-3/ (visited on 09/18/2023).

[10] Harry Deutsch and Oliver Marshall. “Alonzo Church”. In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta and Uri Nodelman. Winter 2023. Metaphysics Research Lab, Stanford University, 2023. URL: https://plato.stanford.edu/archives/win2023/entries/church/.

[11] Chelsea Finn, Pieter Abbeel, and Sergey Levine. *Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks*. July 18, 2017. arXiv: 1703.03400[cs]. URL: http://arxiv.org/abs/1703.03400 (visited on 09/25/2023).

[12] Thomas Gabor et al. “Self-Replication in Neural Networks”. In: *Artificial Life* 28.2 (June 28, 2022), pp. 205–223. ISSN: 1064-5462, 1530-9185. DOI: 10.1162/artl_a_00359. URL: https://direct.mit.edu/artl/article/28/2/205/111793/Self-Replication-in-Neural-Networks (visited on 09/18/2023).

[13] Andy Gill. “Domain-specific Languages and Code Synthesis Using Haskell”. In: *University of Kansas* (Apr. 15, 2014).

[14] W.K. Giloi. “Konrad Zuse’s Plankalkül: the first high-level, ”non von Neumann” programming language”. In: *IEEE Annals of the History of Computing* 19.2 (June 1997), pp. 17–24. ISSN: 10586180. DOI: 10.1109/85.586068. URL: http://ieeexplore.ieee.org/document/586068/ (visited on 09/18/2023).

[15] Kurt Gödel. “On Formally Undecidable Propositions of Principia Mathematica And Related Systems”. Trans. by Meltzer B. In: (1931).

[16] Ben Goertzel and Cassio Pennachin, eds. *Artificial general intelligence.* Cognitive technologies. Berlin ; New York: Springer, 2007. 508 pp. ISBN: 978-3-540-23733-4.

[17] Anand Gopalakrishnan et al. *Unsupervised Learning of Temporal Abstractions with Slot-based Transformers.* Nov. 22, 2022. arXiv: 2203.13573[cs]. URL: http://arxiv.org/abs/2203.13573 (visited on 09/24/2023).

[18] Dakai Guo, this link will open in a new window Link to external site, and Wensheng Yu. "A Comprehensive Formalization of Propositional Logic in Coq: Deduction Systems, Meta-Theorems, and Automation Tactics". In: *Mathematics* 11.11 (2023). Num Pages: 2504 Place: Basel, Switzerland Publisher: MDPI AG, p. 2504. DOI: 10.3390/math11112504. URL: https://www.proquest.com/docview/2824017491/abstract/7553220F76814F76PQ/2 (visited on 09/18/2023).

[19] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997). Publisher: MIT press, pp. 1735–1780.

[20] Timothy Hospedales et al. "Meta-Learning in Neural Networks: A Survey". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.9 (Sept. 2022). Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 5149–5169. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2021.3079209.

[21] F Hutter. *Automated {Machine} {Learning}: {Methods}, {Systems}, {Challenges}*. Ed. by Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. Series Title: The {Springer} {Series} on {Challenges} in {Machine} {Learning}. Cham: Springer International Publishing, 2019. ISBN: 978-3-030-05317-8 978-3-030-05318-5. DOI: 10.1007/978-3-030-05318-5. URL: http://link.springer.com/10.1007/978-3-030-05318-5 (visited on 09/20/2023).

[22] Kazuki Irie et al. *A Modern Self-Referential Weight Matrix That Learns to Modify Itself.* June 17, 2022. DOI: 10.48550/arXiv.2202.05780. arXiv: 2202.05780[cs]. URL: http://arxiv.org/abs/2202.05780 (visited on 10/23/2023).

[23] Kazuki Irie et al. "Going Beyond Linear Transformers with Recurrent Fast Weight Programmers". In: *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc., 2021, pp. 7703–7717. URL: https://proceedings.neurips.cc/paper_files/paper/2021/hash/3f9e3767ef3b10a0de4c256d7ef9805d-Abstract.html (visited on 09/24/2023).

[24] Albert Q Jiang et al. "Thor: Wielding Hammers to Integrate Language Models and Automated Theorem Provers". In: ().

[25] Siran Lei et al. "A Machine Proof System of Point Geometry Based on Coq". In: *Mathematics* 11.12 (June 18, 2023), p. 2757. ISSN: 2227-7390. DOI: 10.3390/math11122757. URL: https://www.mdpi.com/2227-7390/11/12/2757 (visited on 09/18/2023).

[26] Cheuk Ting Li. *An Automated Theorem Proving Framework for Information-Theoretic Results*. July 11, 2022. arXiv: 2101.12370[cs,math]. URL: http://arxiv.org/abs/2101.12370 (visited on 09/18/2023).

[27] Adam Machowczyk and Reiko Heckel. *Graph Rewriting for Graph Neural Networks*. May 29, 2023. arXiv: 2305.18632[cs]. URL: http://arxiv.org/abs/2305.18632 (visited on 09/18/2023).

[28] Erik Meijer, Maarten Fokkinga, and Ross Paterson. "Functional programming with bananas, lenses, envelopes and barbed wire". In: *Functional Programming Languages and Computer Architecture*. Ed. by John Hughes. Red. by Gerhard Goos and Juris Hartmanis. Vol. 523. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 1991, pp. 124–144. ISBN: 978-3-540-54396-1 978-3-540-47599-6. DOI: 10.1007/3540543961_7. URL: http://link.springer.com/10.1007/3540543961_7 (visited on 10/12/2023).

[29] Thomas Miconi et al. *Backpropamine: training self-modifying neural networks with differentiable neuromodulated plasticity*. Feb. 24, 2020. arXiv: 2002.10585[cs]. URL: 2 (visited on 09/20/2023).

[30] Tsendsuren Munkhdalai and Adam Trischler. *Metalearning with Hebbian Fast Weights*. July 12, 2018. arXiv: 1807.05076[cs,stat]. URL: http://arxiv.org/abs/1807.05076 (visited on 09/24/2023).

[31] Vladimir Nasteski. "An overview of the supervised machine learning methods". In: *HORIZONS.B* 4 (Dec. 15, 2017), pp. 51–62. DOI: 10.20544/HORIZONS.B.04.1.17.P05.

[32] Robert A. Nawrocki, Richard M. Voyles, and Sean E. Shaheen. "A Mini Review of Neuromorphic Architectures and Implementations". In: *IEEE Transactions on Electron Devices* 63.10 (Oct. 2016), pp. 3819–3829. ISSN: 0018-9383, 1557-9646. DOI: 10.1109/TED.2016.2598413. URL: http://ieeexplore.ieee.org/document/7549034/ (visited on 09/18/2023).

[33] James O' Neill. *An Overview of Neural Network Compression.* Aug. 1, 2020. arXiv: 2006.03669[cs,stat]. URL: http://arxiv.org/abs/2006.03669 (visited on 09/18/2023).

[34] Alex Nichol, Joshua Achiam, and John Schulman. *On First-Order Meta-Learning Algorithms.* Oct. 22, 2018. arXiv: 1803.02999[cs]. URL: http://arxiv.org/abs/1803.02999 (visited on 09/20/2023).

[35] Donald Perlis. "Languages with self-reference I: Foundations". In: *Artificial Intelligence* 25.3 (Mar. 1985), pp. 301–322. ISSN: 00043702. DOI: 10.1016/0004-3702(85)90075-X. URL: https://linkinghub.elsevier.com/retrieve/pii/000437028590075X (visited on 10/11/2023).

[36] Stanislas Polu and Ilya Sutskever. *Generative Language Modeling for Automated Theorem Proving.* Sept. 7, 2020. arXiv: 2009.03393[cs,stat]. URL: http://arxiv.org/abs/2009.03393 (visited on 09/18/2023).

[37] Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. "Linear Transformers Are Secretly Fast Weight Programmers". In: *Proceedings of the 38th International Conference on Machine Learning.* International Conference on Machine Learning. ISSN: 2640-3498. PMLR, July 1, 2021, pp. 9355–9366. URL: https://proceedings.mlr.press/v139/schlag21a.html (visited on 09/24/2023).

[38] Imanol Schlag, Tsendsuren Munkhdalai, and Jürgen Schmidhuber. *Learning Associative Inference Using Fast Weight Memory.* Feb. 23, 2021. arXiv: 2011.07831[cs]. URL: http://arxiv.org/abs/2011.07831 (visited on 09/24/2023).

[39] J. Schmidhuber. "A 'Self-Referential' Weight Matrix". In: *ICANN '93.* Ed. by Stan Gielen and Bert Kappen. London: Springer London, 1993, pp. 446–450. ISBN: 978-3-540-19839-0 978-1-4471-2063-6. DOI: 10.1007/978-1-4471-2063-6_107. URL: https://mediatum.ub.tum.de/doc/814784/document.pdf (visited on 09/18/2023).

[40] Juergen Schmidhuber. *Godel machine poster*. Dec. 17, 2006. arXiv: `cs.LO/0309048`. URL: `http://arxiv.org/abs/cs/0309048` (visited on 10/01/2023).

[41] Juergen Schmidhuber. *Godel Machines: Self-Referential Universal Problem Solvers Making Provably Optimal Self-Improvements*. Dec. 17, 2006. arXiv: `cs/0309048`. URL: `http://arxiv.org/abs/cs/0309048` (visited on 10/01/2023).

[42] Jürgen Schmidhuber. "An'introspective'network that can learn to run its own weight change algorithm". In: *1993 third international conference on artificial neural networks*. IET, 1993, pp. 191–194.

[43] Jürgen Schmidhuber. "Gödel Machines: Towards a Technical Justification of Consciousness". In: *Adaptive Agents and Multi-Agent Systems II*. Ed. by Daniel Kudenko, Dimitar Kazakov, and Eduardo Alonso. Red. by David Hutchison et al. Vol. 3394. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 1–23. ISBN: 978-3-540-25260-3 978-3-540-32274-0. DOI: `10.1007/978-3-540-32274-0_1`. URL: `http://link.springer.com/10.1007/978-3-540-32274-0_1` (visited on 09/20/2023).

[44] Jürgen Schmidhuber. "Learning complex, extended sequences using the principle of history compression". In: *Neural Computation* 4.2 (1992). Publisher: MIT Press, pp. 234–242.

[45] Jürgen Schmidhuber. "Learning to control fast-weight memories: An alternative to dynamic recurrent networks". In: *Neural Computation* 4.1 (1992). Publisher: MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info . . ., pp. 131–139.

[46] Jürgen Schmidhuber. "Ultimate Cognition à la Gödel". In: *Cognitive Computation* 1.2 (June 2009), pp. 177–193. ISSN: 1866-9956, 1866-9964. DOI: `10.1007/s12559-009-9014-y`. URL: `http://link.springer.com/10.1007/s12559-009-9014-y` (visited on 09/20/2023).

[47] Jürgen Schmidhuber, Jieyu Zhao, and Nicol N Schraudolph. "Reinforcement {Learning} with {Self}-{Modifying} {Policies}". In: *Learning to {Learn}*. Ed. by Sebastian Thrun and Lorien Pratt. Boston, MA: Springer US, 1998, pp. 293–309. ISBN: 978-1-4613-7527-2 978-1-4615-5529-2. DOI: `10.1007/978-1-4615-5529-2_12`. URL: `http://link.springer.com/10.1007/978-1-4615-5529-2_12` (visited on 09/20/2023).

[48] Bas R. Steunebrink and JÃ¼rgen Schmidhuber. “Towards an Actual Gödel Machine Implementation: a Lesson in Self-Reflective Systems”. In: *Theoretical Foundations of Artificial General Intelligence*. Ed. by Pei Wang and Ben Goertzel. Vol. 4. Series Title: Atlantis Thinking Machines. Paris: Atlantis Press, 2012, pp. 173–195. ISBN: 978-94-91216-61-9 978-94-91216-62-6. DOI: `10.2991/978-94-91216-62-6_10`. URL: `https://www.atlantis-press.com/doi/10.2991/978-94-91216-62-6_10` (visited on 10/01/2023).

[49] Geoff Sutcliffe and Christian Suttner. “Evaluating general purpose automated theorem proving systems”. In: *Artificial Intelligence* 131.1 (Sept. 2001), pp. 39–54. ISSN: 00043702. DOI: `10.1016/S0004-3702(01)00113-8`. URL: `https://linkinghub.elsevier.com/retrieve/pii/S0004370201001138` (visited on 10/25/2023).

[50] Cynthia S Symons and Blair T Johnson. “The self-reference effect in memory: a meta-analysis.” In: *Psychological bulletin* 121.3 (1997). Publisher: American Psychological Association, p. 371.

[51] Van-Nhan Tran et al. “Generalization of Forgery Detection With Meta Deepfake Detection Model”. In: *IEEE Access* 11 (2023), pp. 535–546. ISSN: 2169-3536. DOI: `10.1109/ACCESS.2022.3232290`. URL: `https://ieeexplore.ieee.org/document/9999180/` (visited on 10/02/2023).

[52] Joaquin Vanschoren. *Meta-Learning: A Survey.* Oct. 8, 2018. arXiv: `1810.03548[cs,stat]`. URL: `http://arxiv.org/abs/1810.03548` (visited on 09/20/2023).

[53] Daniel Whalen. *Holophrasm: a neural Automated Theorem Prover for higher-order logic.* Aug. 9, 2016. arXiv: `1608.02644[cs]`. URL: `http://arxiv.org/abs/1608.02644` (visited on 09/18/2023).

[54] Xingqi Zou et al. “Breaking the von Neumann bottleneck: architecture-level processing-in-memory technology”. In: *Science China Information Sciences* 64.6 (June 2021), p. 160404. ISSN: 1674-733X, 1869-1919. DOI: `10.1007/s11432-020-3227-1`. URL: `https://link.springer.com/10.1007/s11432-020-3227-1` (visited on 09/18/2023).

Index