APSIM meets TensorFlow: Optimising sequential management decisions via deep reinforcement learning

Yuji Saikai

School of Mathematics and Statistics, the University of Melbourne

Abstract

APSIM (Agricultural Production Systems sIMulator) is capable of creating high-fidelity simulations of diverse agricultural systems. It is used by both academics and practitioners for many different purposes, one of which is optimisation of management decisions. When addressing realistic dynamic optimisation problems, an exhaustive search for the best practice becomes computationally impossible. Deep reinforcement learning is a viable approach to solving such challenging problems. APSIMTF is a software bridge between APSIM and TensorFlow, enabling researchers to implement their own deep reinforcement learning algorithms in APSIM environments. This paper describes the design principle of APSIMTF, demonstrates how it works using a scenario of optimal irrigation scheduling, and discusses existing challenges, suggestions, and outlook for the further development.

Keywords: APSIM, TensorFlow, deep learning, reinforcement learning, C#

1 Introduction

Over the past three decades, APSIM (Agricultural Production Systems sIMulator) as a farming systems modelling framework has dramatically improved its capability to simulate a wide range of complex agricultural systems [1]. It is now used across the world for examination of soil sustainability processes, evaluation of resource use and efficiency, environmental characterisation, plant breeding, whole farm approaches, and understanding mixed crop-livestock enterprises, to name a few [2]. Its high fidelity to systems of interest gives modellers confidence in their models [3] and enables *in silico* experimentation, which is extremely valuable when experiments are either impossible or infeasible in real-world environments. As noted in Ittersum and Donatelli [4], such a high-fidelity simulator is particularly useful for optimisation of management practices (i.e., search for the best practices through trial and error) in both scientific and practical uses such as investigation of yield potential [3, 5] and development of decision support systems [6, 7].

Despite the potential for considerable benefits, currently only simple static optimisation problems are addressed using APSIM. For example, in a study of yield potential [5], only five wheat varieties are considered as candidates for the best variety, while other management choices, including sowing and fertiliser rules, are fixed and implicitly assumed optimal. As another example, in Rees et al. [3], only three rates of additional nitrogen application or only a handful of different sowing timing and maturing speeds are investigated as candidates, while farmers' other practices are explicitly assumed to be the best practices.

In reality, many of the assumed optimal practices are not fixed but also part of farmers' choices. Inclusion of these choices as explicit decision variables will significantly amplify the scale of optimisation problems because the total number of candidates exponentially increases as types of practices increases. In addition, farmers' decisions are made sequentially over the growing season [8], and such dynamic structure causes another source of exponential increase in the scale of optimisation problems because of the combinatorial nature of sequential decisions over time. In APSIM simulations, it is common to use daily decision frequency. So, for example, with 200 days of growing season, even a binary daily decision (e.g., irrigate or not) creates $2^{200} \approx 10^{60}$ candidates for the best combination, which makes an exhaustive search impossible.

In principle, this class of dynamic optimisation problems can be solved through dynamic programming *under the special conditions* [9–11]. In realistic agricultural systems, however, there is at least one condition that is almost impossible to satisfy — dynamics is explicitly known. Known dynamics here means that decision makers know and can mathematically formulate how today's decision basis (e.g., plant health and soil properties) will change tomorrow given their decisions made today. Even in APSIM simulations where each component process is explicitly specified, it is hard and essentially impossible to write down explicit expressions of the entire dynamics.

Reinforcement learning is a subfield of machine learning and designed to solve the same class of dynamic optimisation problems based on decision makers' choice experiences *under weaker conditions* than those for dynamic programming [12]. As a form of machine learning, reinforcement learning relies on data that encodes key information of decision makers' experiences. The weaker conditions allow decision makers to address dynamic optimisation problems without knowing dynamics, which is considerable advantage of reinforcement learning over dynamic programming. In agriculture, while some work has been done in agricultural engineering (e.g., learning optimal drone navigation [13, 14]), there exist only a few papers on sequential management optimisation [15, 16].

When applying reinforcement learning to realistic agricultural optimisation problems, the enormous complexity of agricultural systems is a significant challenge to overcome. Agricultural systems typically consist of ecological factors and human management factors. While the former include evolution of plant status and soil properties as well as their interaction, the latter are external interventions into such complex ecological systems. Therefore, optimal management decisions need to reflect the ecological complexity, or equivalently, an optimal decision rule is a complex function mapping each of possible decision basis to an optimal decision. Without assuming the knowledge of dynamics, reinforcement learning must learn a complex decision rule from data. This is a fundamental challenge faced when applying reinforcement learning to optimisation problems with real-world complexity [17]. A solution to the challenge is to take advantage of the representational power of deep learning [18], which is capable of learning complex functions from data. Hence, deep reinforcement learning is currently a promising approach to this fundamental challenge [17, 19].

When implementing deep learning algorithms, it is de facto standard to use dedicated software libraries so that researchers can significantly reduce coding work and focus on modelling. Among several established options (e.g., PyTorch and Theano), TensorFlow has been the most popular library widely used by both academics and practitioners [20, 21]. Given the need for deep reinforcement learning in APSIM environments, therefore, it would be of great service to develop a software bridge between APSIM and TensorFlow.

This paper first describes the current status of APSIMTF, a stripped-down version of ApsimX (the next generation of APSIM), with particular emphasis on software design for TensorFlow integration into ApsimX. Next, the paper demonstrates how the software works using a scenario of optimal irrigation scheduling for wheat production in Australia. The demonstration provides a concrete example of the abstract design concept and code structure described in the previous section. The section also presents details of the learned irrigation scheduling, a stochastic decision rule that applies 0mm, 20mm, 40mm, 60mm, or 80mm of water each day in order to maximise yield at the end of the season. Finally, the paper discusses challenges, suggestions, and outlook for the further development. All the code, results, APSIM simulation file, and Visual Studio solution files prepared by the author are available in the GitHub repository (https://github.com/ysaikai/APSIMTF).

2 Software

2.1 Design

Reinforcement learning requires interaction between a learning agent and the environment [12]. Specifically, given a decision basis called "state", Agent takes an action following the decision rule called "policy". Environment in turn gives back to Agent an immediate reward and the next state, in which Agent is going to be (Fig.1). In other words, Environment is a passive mechanism that responds to actions Agent actively takes.

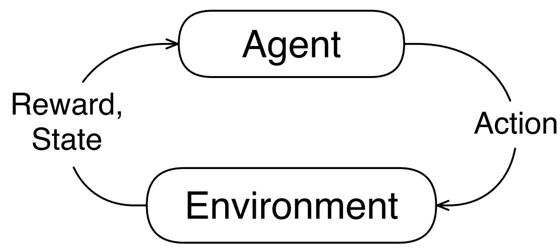

Figure 1: The agent–environment interaction in reinforcement learning.

As a result, to construct the interactive system in software, it is natural to maintain two separate classes of code, an environment class and an agent class, where the environment class is essentially a static function that takes the current state and action as arguments and returns a reward and the next state. All the rest of implementation, including a learning algorithm and learned policy, is coded in the agent class (Fig.2). Under this modularisation, the environment class may be even a black-box function and exogenously given. This is indeed the case in many of the existing reinforcement learning applications using, for example, OpenAI Gym [22]. However, the approach is difficult to achieve with APSIM environments unless making substantial modifications of the original code.

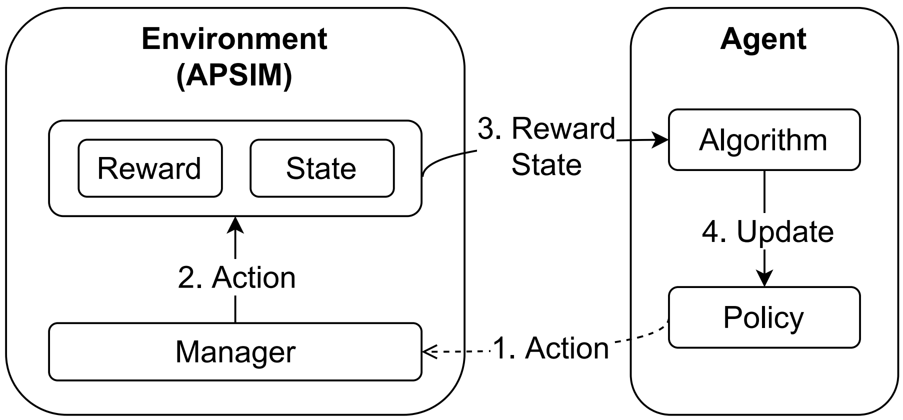

Figure 2: A natural software design for the interactive system. The arrows represent key information flows. The bottom dotted arrow (#1) indicates a problematic process.

The challenge stems from two characteristics of the software architecture of APSIM. The first is the way APSIM allows modellers to implement custom farm management rules. It is basically done through its *Manager* module [23], which allows modellers to write management rules in an external text file and dynamically compiles the code into the software upon running the corresponding simulations. Although very useful for modellers spared the trouble of modifying the source code only for custom management rules, it is not ideal for the agent-environment interaction in reinforcement learning, which often requires a large number of communication, because the agent class needs to send actions or specify a policy through the slow text file I/O process. The second source of the challenge is the lack of ways to communicate with the *Manager* module on a daily basis, which is the standard time step in APSIM simulations. Since external management rules are usually embedded in .apsim or .apsimx files, which are read in and compiled only once before every simulation, there is not an easy way to dynamically update management rules within a simulation even if a single simulation may go over hundreds of days. This may be acceptable for some reinforcement learning algorithms (e.g., Monte Carlo methods) that do not update policies at every time step but do so at every episode, which is a meaningful unit consisting of multiple steps and usually corresponds to a growing season (or year) in APSIM environments. However, other algorithms (e.g., temporal difference methods) update policies within an episode.

A way around to circumvent the challenge is to create a static management rule that simply asks *Policy* which action to take at every time step (#1). Then, *Policy* returns an action (#2), and *Manager* simply relays it (#3). This way, there is no need to dynamically change the management rule as it always sends to *Policy* the same query—"What to do?" (Fig.3). To make *Policy* accessible from *Manager*, the environment class and the agent class are contained as sub-classes in a single APSIM class.

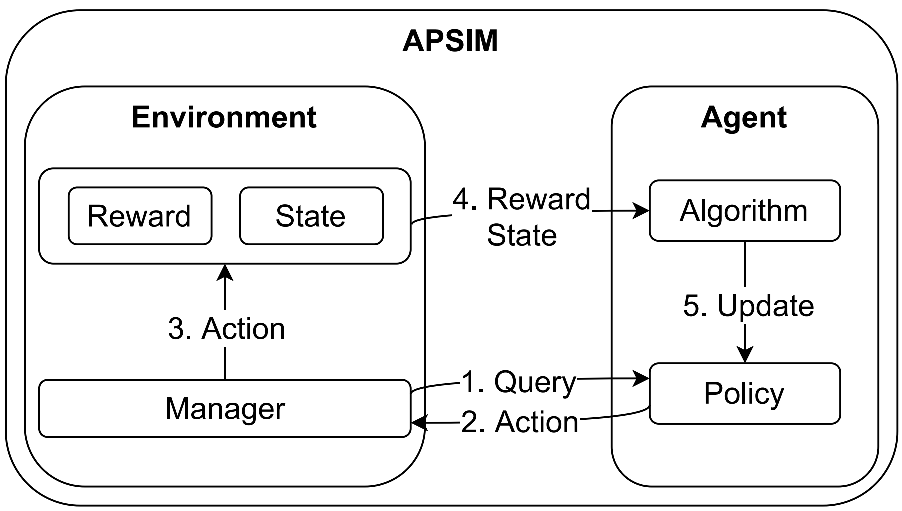

Figure 3: A proposed software design. The arrows represent key information flows.

2.2 Code

In approximating key functions (e.g., value functions, policies, and environment dynamics), deep reinforcement learning relies on deep learning, whose implementation is significantly facilitated by dedicated software libraries such as TensorFlow. TensorFlow can be used through APIs, which are provided in many languages including Python, JavaScript, and C++ [24]. While Python is by far the most popular language, for tight integrate of TensorFlow into APSIM, it is almost necessary to use C# because ApsimX (the open-source version of APSIM) is written in C#. To this end, the most developed option is TensorFlow.NET, a collection of .NET Standard bindings for TensorFlow [25]. Since it follows .NET Standard 2.0, it is readily used along with ApsimX, which is targeted on .NET Framework 4.6.1.

Despite the relevance of yield in agricultural production, it is surprisingly difficult to access a variable containing a yield value in the ApsimX source code. While yield is readily found in the default SQLite database, for performance, it is ideal to have direct access to a variable in memory. Besides the database I/O processes, yield and related quantities are used in many processes in APSIM, but they are somewhat complex and modification is required for direct access. One of the simplest ways to capture a yield value is to insert the following two lines at Line 228 in Report.cs.

```
if (columns[i].Name == "Yield")
    Program.yield = (float)Convert.ToDouble(valuesToWrite.Last());
```

As of October 1, 2020, it should be placed right below the following line at Line 227:

```
valuesToWrite.Add(columns[i].GetValue(groupIndex));
```

For this to work, it is necessary to have the following:

- Program.yield is declared as a public float variable,
- Reporting variables specified in the .apsimx file include one referred to as “Yield” such as [Wheat].Grain.Total.Wt*10 as Yield, and
- Reporting frequency specified in the .apsimx file contains only “Harvesting” such as [Wheat].Harvesting.

In addition to capturing yield, there are three files: Main.cs, PolicyNet.cs, and IrrigationPolicy.cs loosely corresponding to respectively Environment, Agent, and Manager in Fig.3. Management.cs contains the code used for the *Manager* module. Once the code is copied into the module, the file can be discarded. Daily interaction with *Policy* is implemented by subscribing the “DoManagement” event. Under “EndOfDay” event, today’s action and the next state are appended to the corresponding public variables. PolicyNet.cs contains the code to build, evaluate, and update a neural network using TensorFlow.NET. Details of the code depends on specific neural network architecture and learning algorithm in individual applications. Since TensorFlow.NET significantly facilitates porting code written in Python to C#, researchers may first search for suitable models in a large collection of Python code examples and then transfer them into C# using TensorFlow.NET. Main.cs is used as a replacement of the file of the same name in ApsimX and, as such, responsible for Main method (i.e., an entry point in C# applications), console inputs, parameter specifications, variable declarations, data storage, outputs, and overall control flow. The neural network defined in PolicyNet.cs is also instantiated in Main.cs.

2.3 Build

APSIMTF is a stripped-down version of ApsimX, retaining only necessary modules for deep reinforcement learning as a console application. As a result, when ApximX is downloaded from the official GitHub repository [26] and unzipped, most of the files and folders can be removed. Specifically, APSIMTF needs to keep only Models and APSIM.Shared folders. Main.cs and PolicyNet.cs must be placed in Models folder. In addition, as mentioned in the previous section, remember that Report.cs found in ./Models/Report folder must be modified to capture a yield value. A simulation file (.apsimx) and weather file (.met) must be present in the same folder, and users may use their own ones by changing the corresponding parameters in Main.cs. Finally, to build the source code via Visual Studio, an appropriate solution file (.sln) must also be present in the same folder. The author's solution file may be used as a template and modified for specific applications. APSIMTF has been tested on 64-bit Windows 10 machines with Visual Studio 2019, ApsimX (as of October 1, 2020), SciSharp.TensorFlow.Redist (v2.3.1) and TensorFlow.NET (v0.21.0), the last two of which can be installed via NuGet.

3 Demonstration

3.1 Scenario

Since optimising water resource management is one of the most pressing problmes in agriculture [16, 27, 28], this research scenario investigates an optimal irrigation scheduling through deep reinforcement learning facilitated by APSIMTF. A simulated production system is irrigated winter wheat in Dalby, Queensland, Australia. Dalby is chosen simply because ApsimX provides most weather data for the region. The weather file contains data for Year 1900-2000 except 22 of them in which simulations do not run properly. (Specifically, the years excluded are 1906, 1908, 1911, 1915, 1917, 1918, 1924, 1929, 1933, 1937, 1941, 1943, 1944, 1946, 1954, 1960, 1962, 1971, 1972, 1984, 1991, and 1994.) In total, there are $79 = 101 - 22$ possible years, from which a single year randomly realises at each simulation run.

Given the nature of the task (irrigation scheduling optimisation), four state variables are selected from the numerical variables calculated in APSIM. These are day of the year (Day), leaf area index (LAI), extractable soil water (ESW), and cumulative irrigation amount (CuIrrig). Day of the year provides timing information, while an cumulative irrigation amount provides a summary of the decision sequence. ESW here is the sum of separate ESW numbers APSIM calculates at seven different levels of soil depth (cm): 0-15, 15-30, 30-60, 60-90, 90-120, 120-150 and 150-180. Initial soil water is set at 50% full in all simulations. All the rest of configurations is left unchanged from the ones specified in Wheat.apsim, which is provided as an example simulation by ApsimX.

A irrigation policy is a function that prescribes one of five irrigation amounts given a set of four state values each day. The candidate amounts are 0mm, 20mm, 40mm, 60mm, and 80mm per day. The objective of study is to learn from repeated trials a policy that maximises expected wheat yield in a randomly realised year. Learning starts at sowing and ends at harvesting. This implies that duration of learning may vary at each run because year (and weather) realisation is random and days of sowing and harvesting may vary in different years.

3.2 Method

A class of function to learn is a neural network with five hidden layers. The specific architecture consists of the first and fifth layers with 128 nodes, the second and fourth layers with 256 nodes, and the third layer with 512 nodes as well as a single bias node at each layer (Fig.4). With this architecture, the total number of parameters to estimate is:

$$\begin{aligned} 329,989 = &(4 \times 128) + (128 \times 256 + 256) + (256 \times 512 + 512) + \\ &(512 \times 256 + 256) + (256 \times 128 + 128) + (128 \times 5 + 5). \end{aligned}$$

The activation functions are ReLU (rectified linear unit) [29] at each hidden layer and the standard softmax at the output layer.

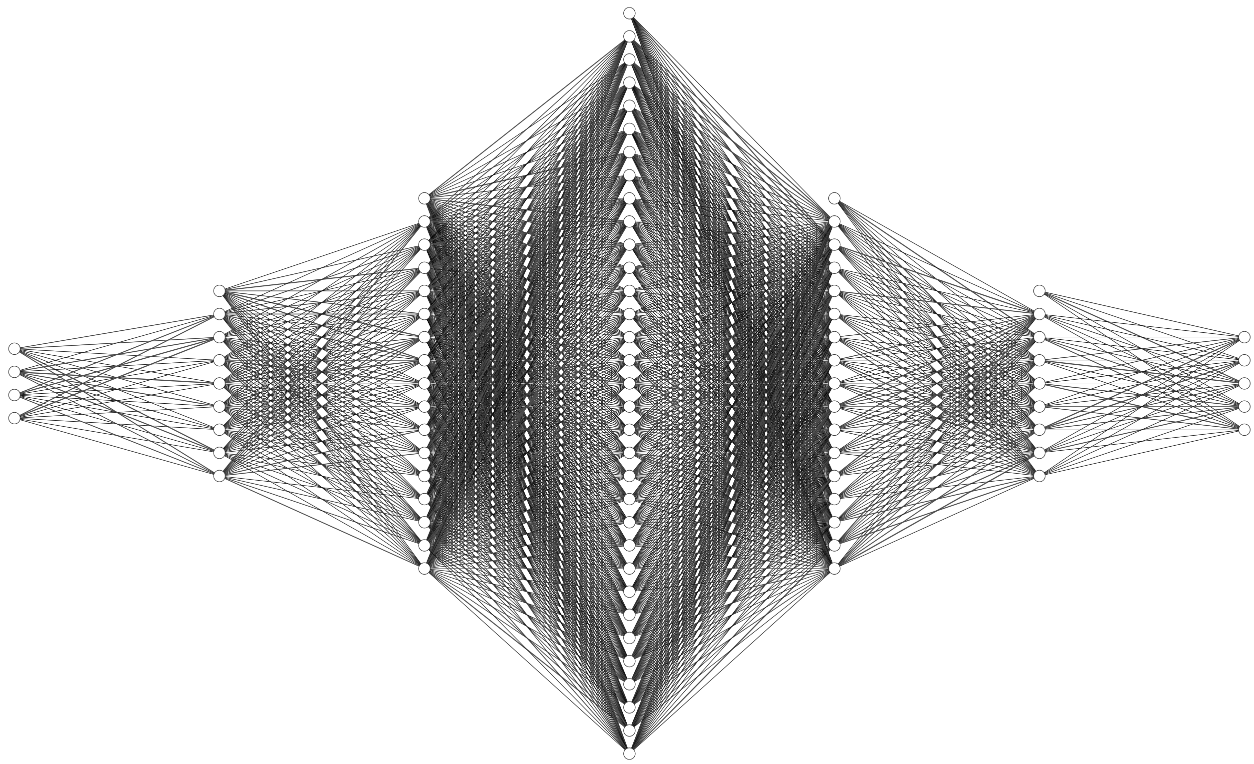

Figure 4: A network diagram indicating four state variables at the input layer and five candidate actions at the output layer. Due to the space restriction, only 1/16 of the total number of nodes at each hidden layer is drawn. A bias node is also drawn at each hidden layer.

The learning algorithm is based on a classic policy gradient method called REINFORCE [30]. A learning rate is set to 0.001. Due to the weather stochasticity created by random years, to evaluate the online performance of the policy as learning progresses, moving average of order 50 is adopted. That is, the performance after each year is the average yield over the past 50 random years. Given the total 79 possible years, the order 50 is arbitrary but reasonable to obtain informative performance estimates while avoiding excessive smoothing. Note that the choice of REINFORCE for the learning algorithm is only for demonstration of APSIMTF. If researchers pursue higher performance, there exist more sophisticated algorithms such as actor-critic methods [12].

Due to the multiple sources of randomness (i.e., year, learned policy, and parameter initialisation), the study examined a set of 10 random seeds (1,2,...10). Note that 10 random seeds implies 10 independent sets of learning. Therefore, the order of seeds is irrelevant and, to replicate the results, researchers may start with any seed or examine only specific seeds. For each seed, learning takes place over 2000 episodes (i.e., 2000 random years), and records the best learned policy that has the highest moving average of yield at any point over 2000 episodes. Immediately after 2000 episodes without seed initialisation, the best policy is tested in each of 79 years, which means that the test results also depend on the seed as the best policy is itself stochastic. Also note that the "best" policy is determined by the moving average of yield, which depends on the seed and may not be representative of the average yield over 79 distinct years. The criterion is justified by the fact that, in contrast to simulations, researchers cannot run separate tests in every possible weather pattern in real-world learning.

3.3 Results

The best policy has 5,856 (kg/ha) of moving average at Episode 1455 under random seed 8. With this policy, the average yield over 79 years is 5,860 (kg/ha). While it is hard to describe a stochastic policy implemented in stochastic environments, Table 1 in Appendix may provide an insight into the learned policy. Aligned with our intuition, the probability of no irrigation ($p(0)$) gradually decreases as ESW decreases, while it jumps up after any positive amount of irrigation. However, the extent to which these general patterns hold varies depending on production stages (Day), plant health (LAI), and past actions (CuIrrig).

For performance comparison, benchmark results are obtained by simulating the same set of 79 years with an irrigation policy called "Automatic irrigation based on water deficit", which is provided by ApsimX. The following figure plots 79 yields from the learned policy against the corresponding yields from the automatic irrigation. While both policies have similar yields in many years, the learned policy performs significantly better in some years. Hence, the average yield of the automatic irrigation over 79 years is 5,722 (kg/ha), which is 138 kg/ha lower than that of the learned policy. Also note that information requirement is different between two policies. The learned policy is based on LAI and ESW estimates, while the automatic policy is based on water deficit that is calculated using potential and currently available soil water.

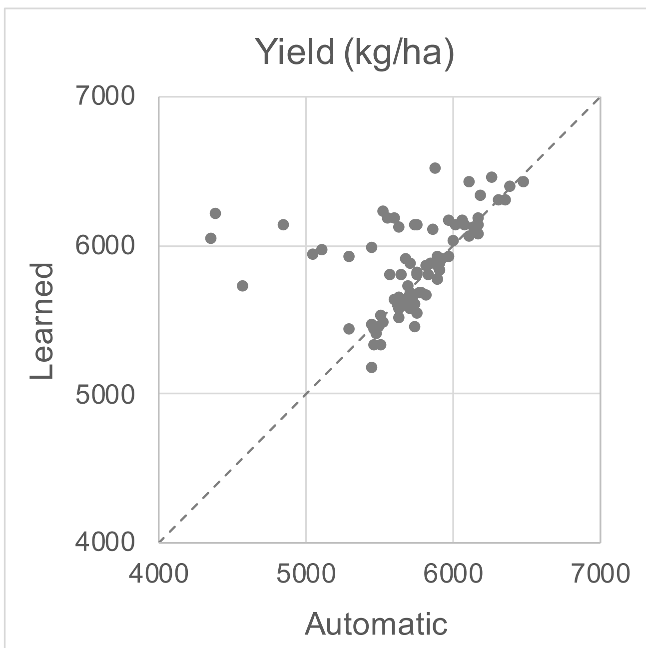

Figure 5: A scatter plot of yield obtained from the automatic irrigation and the learned policy.

4 Discussion

The development of APSIMTF has reached a milestone where it achieves basic integration of TensorFlow into APSIM with the minimum modification of the original code. Using APSIMTF, researchers and practitioners can readily implement their deep reinforcement learning algorithms in APSIM environments. For those who have experience in TensorFlow and reinforcement learning, the author's code example should provide them with a useful starting point. Regarding the computational performance, on modern laptops, APSIMTF can handle tens of thousands of episodes and process each episode for approximately two seconds. However, there is certainly room for improvement in terms of user interface, processing speed and memory management.

Currently, instead of providing graphical user interface (GUI), APSIMTF requires some amount of coding for individual applications, which is common in machine learning practice. On the one hand, GUI would be certainly helpful for environmental and agricultural modellers with little coding experience. On the other hand, a class of deep learning models is very general and its specification may easily involve hundreds of architectural decisions and hyperparameters. In addition, reinforcement learning algorithms are also diverse, each of which requires specific implementation. Therefore, a one-size-fits-all GUI would likely be cluttered yet still inflexible. The balance between ease of use and flexibility is an open question.

To increase the processing speed, one of the measures is to remove all the database I/O process. For its rich functionality, ApsimX constantly accesses a database throughout even a single simulation. Although APSIMTF itself does not rely on any database I/O, it is still in the background process. Complete removal of database process, thus, could meaningfully increase the processing speed.

While APSIMTF establishes smooth interaction between a learning agent and the APSIM environment within an episode, the environment should be independent between episodes for tasks with finite horizon (i.e., each episode ends after a finite number of steps). However, currently the memory usage steadily increases as the number of episodes grows. It would be ideal to reset the environment and release most of the memory used by the APSIM processes.

All the improvement critically depends on effective communication with other developers and users, in particular with the current and future ApsimX developers. Thus far, the development of APSIMTF and its description in this paper are entirely based on the author's understanding of ApsimX. Without doubt, there are misunderstandings of its design and details in code. It is author's hope that the publication of this paper will initiate a discourse in the APSIM community.

References

[1] Dean P Holzworth, Neil I Huth, Peter G deVoil, Eric J Zurcher, Neville I Herrmann, Greg McLean, Karine Chenu, Erik J van Oosterom, Val Snow, and Chris Murphy. "APSIM–evolution towards a new generation of agricultural systems simulation". In: *Environmental Modelling & Software* 62 (2014), pp. 327–350.

[2] Dean Holzworth, Neil I Huth, Justin Fainges, H Brown, E Zurcher, Rogerio Cichota, Shaun Verrall, Neville I Herrmann, B Zheng, and V Snow. "APSIM Next Generation: overcoming challenges in modernising a farming systems model". In: *Environmental Modelling & Software* 103 (2018), pp. 43–51.

[3] Harm van Rees, Tim McClelland, Zvi Hochman, Peter Carberry, James Hunt, Neil Huth, and Dean Holzworth. "Leading farmers in South East Australia have closed the exploitable wheat yield gap: Prospects for further improvement". In: *Field Crops Research* 164 (2014), pp. 1–11.

[4] M.K. van Ittersum and M. Donatelli. "Modelling cropping systems—highlights of the symposium and preface to the special issues". In: *Modelling Cropping Systems: Science, Software and Applications* 18.3 (2003), pp. 187–197.

[5] Zvi Hochman, David Gobbett, Heidi Horan, and Javier Navarro Garcia. "Data rich yield gap analysis of wheat in Australia". In: *Field Crops Research* 197 (2016). Publisher: Elsevier, pp. 97–106.

[6] Héctor Cancela, Andrew Higgins, Adela Pagès-Bernaus, and Lluis Miquel Plà-Aragonès. “Prologue – BigData and DSS in agriculture”. In: *Computers and Electronics in Agriculture* 161 (2019), pp. 1–3.

[7] Zhaoyu Zhai, José Fernán Martínez, Victoria Beltran, and Néstor Lucas Martínez. “Decision support systems for agriculture 4.0: Survey and challenges”. In: *Computers and Electronics in Agriculture* 170 (2020), p. 105256.

[8] Dean P. Holzworth, Val Snow, Sander Janssen, Ioannis N. Athanasiadis, Marcello Donatelli, Gerrit Hoogenboom, Jeffrey W. White, and Peter Thorburn. “Agricultural production systems modelling and software: Current status and future prospects”. In: *Environmental Modelling & Software* 72 (2015), pp. 276–286.

[9] Richard Bellman. “Dynamic programming”. In: *Science* 153.3731 (1966). Publisher: American Association for the Advancement of Science, pp. 34–37.

[10] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming.* John Wiley & Sons, 1994.

[11] Dimitri P Bertsekas, Dimitri P Bertsekas, Dimitri P Bertsekas, and Dimitri P Bertsekas. *Dynamic programming and optimal control.* Vol. 1. Issue: 2. Athena scientific Belmont, MA, 1995.

[12] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction.* 2nd ed. Cambridge, MA: MIT press, 2018.

[13] Zichen Zhang, Jayson Boubin, Christopher Stewart, and Sami Khanal. “Whole-Field Reinforcement Learning: A Fully Autonomous Aerial Scouting Method for Precision Agriculture”. In: *Sensors* 20.22 (2020). Publisher: Multidisciplinary Digital Publishing Institute, p. 6585.

[14] Jayson Boubin, John Chumley, Christopher Stewart, and Sami Khanal. “Autonomic computing challenges in fully autonomous precision agriculture”. In: 2019 IEEE International Conference on Autonomic Computing (ICAC). IEEE, 2019, pp. 11–17.

[15] F Garcia. “Use of reinforcement learning and simulation to optimize wheat crop technical management”. In: Modsim’99 International congress on modelling and simulation. 1999.

[16] Lijia Sun, Yanxiang Yang, Jiang Hu, Dana Porter, Thomas Marek, and Charles Hillyer. “Reinforcement Learning Control for Water-Efficient Agricultural Irrigation”. In: *2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC).* 2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC). 2017, pp. 1334–1341.

[17] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, and Georg Ostrovski. “Human-level control through deep reinforcement learning”. In: *nature* 518.7540 (2015). Publisher: Nature Publishing Group, pp. 529–533.

[18] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *Nature* 521.7553 (2015), pp. 436–444.

[19] Yuxi Li. “Deep reinforcement learning: An overview”. In: *arXiv preprint arXiv:1701.07274* (2017).

[20] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, and Michael Isard. "TensorFlow: A System for Large-Scale Machine Learning". In: 12th USENIX Symposium on Operating Systems Design and Implementation. 2016, pp. 265–283.

[21] Google. *TensorFlow*. 2015. URL: https://tensorflow.org.

[22] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. "Openai gym". In: *arXiv preprint arXiv:1606.01540* (2016).

[23] APSIM Initiative. *Manager 2*. 2020. URL: https://www.apsim.info/documentation/model-documentation/infrastructure-and-management-documentation/manager-2/.

[24] Google. *API Documentation — TensorFlow Core*. URL: https://www.tensorflow.org/api_docs (visited on 12/31/2020).

[25] Haiping Chen. *TensorFlow.NET*. 2018. URL: https://github.com/SciSharp/TensorFlow.NET.

[26] APSIM Initiative. *ApsimX*. 2020. URL: https://github.com/APSIMInitiative/ApsimX.

[27] Ni Zhou. "Intelligent Control of Agricultural Irrigation Based on Reinforcement Learning". In: *Journal of Physics: Conference Series* 1601.4 (2020).

[28] Guido Maria Bazzani. "An integrated decision support system for irrigation and water policy design: DSIRR". In: *Environmental Modelling & Software* 20.2 (2005). Publisher: Elsevier, pp. 153–163.

[29] Vinod Nair and Geoffrey E Hinton. "Rectified linear units improve restricted boltzmann machines". In: ICML. 2010.

[30] Ronald J Williams. "Simple statistical gradient-following algorithms for connectionist reinforcement learning". In: *Machine learning* 8.3 (1992). Publisher: Springer, pp. 229–256.

Appendix

Table 1: Sequence of realised states and actions taken under the learned policy in 2000. CuIrrig indicates cumulative irrigation amount, while $p(a)$ indicates the probability of applying a amount of water following the policy. The resulting yield is 6,029 (kg/ha).

Day	LAI	ESW	CuIrrig	Action	p(0)	p(20)	p(40)	p(60)	p(80)
163	0.00	209	0	0	0.42	0.19	0.14	0.14	0.11
164	0.00	237	0	0	0.53	0.16	0.11	0.11	0.08
165	0.00	238	0	0	0.53	0.16	0.11	0.11	0.08
166	0.00	235	0	0	0.51	0.17	0.12	0.12	0.09
167	0.00	234	0	40	0.50	0.17	0.12	0.12	0.09
168	0.00	266	40	0	0.80	0.09	0.05	0.04	0.02
169	0.00	263	40	0	0.78	0.10	0.05	0.04	0.03
170	0.00	261	40	0	0.76	0.10	0.06	0.05	0.03
171	0.02	259	40	0	0.75	0.11	0.06	0.05	0.03
172	0.03	258	40	0	0.73	0.11	0.06	0.06	0.04
173	0.03	256	40	40	0.71	0.12	0.07	0.06	0.04
174	0.03	288	80	0	0.95	0.03	0.01	0.01	0.00
175	0.03	285	80	0	0.93	0.04	0.01	0.01	0.00
176	0.03	282	80	0	0.92	0.05	0.02	0.01	0.01
177	0.03	279	80	0	0.90	0.06	0.02	0.02	0.01
178	0.04	277	80	0	0.88	0.06	0.03	0.02	0.01
179	0.04	275	80	0	0.87	0.07	0.03	0.02	0.01
180	0.05	274	80	0	0.86	0.07	0.03	0.02	0.01
181	0.05	273	80	40	0.85	0.07	0.03	0.03	0.01
182	0.06	309	120	0	1.00	0.00	0.00	0.00	0.00
183	0.07	306	120	0	0.99	0.00	0.00	0.00	0.00
184	0.07	305	120	0	0.99	0.01	0.00	0.00	0.00
185	0.08	302	120	0	0.99	0.01	0.00	0.00	0.00
186	0.10	299	120	0	0.99	0.01	0.00	0.00	0.00
187	0.11	297	120	0	0.99	0.01	0.00	0.00	0.00
188	0.13	295	120	0	0.98	0.01	0.00	0.00	0.00
189	0.15	294	120	0	0.98	0.01	0.00	0.00	0.00
190	0.17	293	120	0	0.97	0.02	0.01	0.00	0.00
191	0.18	292	120	0	0.97	0.02	0.01	0.00	0.00
192	0.20	290	120	0	0.96	0.02	0.01	0.00	0.00
193	0.22	289	120	0	0.96	0.03	0.01	0.01	0.00
194	0.26	292	120	0	0.97	0.02	0.01	0.00	0.00
195	0.29	291	120	20	0.96	0.03	0.01	0.00	0.00
196	0.31	308	140	0	0.99	0.00	0.00	0.00	0.00
197	0.34	306	140	0	0.99	0.01	0.00	0.00	0.00
198	0.36	304	140	0	0.99	0.01	0.00	0.00	0.00
199	0.38	301	140	0	0.99	0.01	0.00	0.00	0.00
200	0.41	298	140	0	0.99	0.01	0.00	0.00	0.00
201	0.44	295	140	0	0.98	0.01	0.00	0.00	0.00
202	0.48	293	140	0	0.97	0.02	0.01	0.00	0.00
203	0.53	290	140	0	0.96	0.02	0.01	0.00	0.00
204	0.61	289	140	0	0.95	0.03	0.01	0.01	0.00
205	0.69	287	140	0	0.93	0.04	0.01	0.01	0.00
206	0.77	285	140	0	0.91	0.05	0.02	0.01	0.01
207	0.83	283	140	20	0.89	0.06	0.02	0.02	0.01
208	0.92	300	160	0	0.99	0.01	0.00	0.00	0.00
209	1.00	298	160	0	0.99	0.01	0.00	0.00	0.00
210	1.06	296	160	0	0.98	0.01	0.00	0.00	0.00
211	1.09	293	160	0	0.98	0.02	0.00	0.00	0.00
212	1.12	291	160	0	0.96	0.02	0.01	0.00	0.00
213	1.25	288	160	0	0.95	0.03	0.01	0.01	0.00
214	1.42	286	160	0	0.92	0.04	0.02	0.01	0.01
215	1.54	284	160	0	0.89	0.06	0.02	0.02	0.01
216	1.63	281	160	0	0.87	0.07	0.03	0.02	0.01
217	1.72	278	160	0	0.84	0.08	0.04	0.03	0.02
218	1.80	275	160	0	0.82	0.09	0.04	0.04	0.02
219	1.89	273	160	0	0.80	0.09	0.05	0.04	0.02
220	2.05	271	160	0	0.78	0.10	0.05	0.04	0.03
221	2.20	268	160	0	0.76	0.10	0.06	0.05	0.03
222	2.38	265	160	0	0.74	0.11	0.06	0.06	0.03
223	2.51	261	160	0	0.70	0.12	0.07	0.06	0.04
224	2.61	257	160	0	0.67	0.13	0.08	0.07	0.05
225	2.68	253	160	0	0.63	0.14	0.09	0.08	0.06
226	2.75	249	160	20	0.60	0.15	0.10	0.09	0.06
227	2.92	266	180	0	0.75	0.11	0.06	0.05	0.03
228	3.08	266	180	60	0.74	0.11	0.06	0.05	0.03
229	3.16	297	240	0	0.99	0.01	0.00	0.00	0.00
230	3.21	293	240	0	0.99	0.01	0.00	0.00	0.00
231	3.24	288	240	0	0.98	0.01	0.00	0.00	0.00
232	3.30	285	240	0	0.97	0.02	0.01	0.00	0.00
233	3.37	281	240	0	0.94	0.03	0.01	0.01	0.00
234	3.45	277	240	0	0.89	0.06	0.02	0.02	0.01
235	3.51	272	240	0	0.84	0.08	0.04	0.03	0.02
236	3.57	267	240	40	0.80	0.09	0.05	0.04	0.02
237	3.65	301	280	0	1.00	0.00	0.00	0.00	0.00

Day	LAI	ESW	CuIrrig	Action	p(0)	p(20)	p(40)	p(60)	p(80)
238	3.73	297	280	0	0.99	0.00	0.00	0.00	0.00
239	3.83	292	280	0	0.99	0.01	0.00	0.00	0.00
240	3.96	286	280	0	0.99	0.01	0.00	0.00	0.00
241	4.04	280	280	0	0.97	0.02	0.01	0.00	0.00
242	4.11	277	280	0	0.95	0.03	0.01	0.01	0.00
243	4.17	273	280	0	0.90	0.05	0.02	0.02	0.01
244	4.22	268	280	80	0.85	0.08	0.04	0.03	0.02
245	4.30	300	360	0	1.00	0.00	0.00	0.00	0.00
246	4.37	294	360	0	1.00	0.00	0.00	0.00	0.00
247	4.44	292	360	0	1.00	0.00	0.00	0.00	0.00
248	4.49	286	360	0	1.00	0.00	0.00	0.00	0.00
249	4.62	281	360	0	0.99	0.01	0.00	0.00	0.00
250	4.75	274	360	0	0.99	0.01	0.00	0.00	0.00
251	4.84	268	360	0	0.97	0.02	0.01	0.00	0.00
252	4.93	262	360	0	0.93	0.04	0.02	0.01	0.00
253	5.01	256	360	0	0.87	0.07	0.03	0.02	0.01
254	5.07	251	360	0	0.83	0.08	0.04	0.03	0.02
255	5.09	244	360	60	0.78	0.10	0.05	0.04	0.03
256	5.06	281	420	0	1.00	0.00	0.00	0.00	0.00
257	5.04	275	420	0	1.00	0.00	0.00	0.00	0.00
258	5.02	269	420	0	0.99	0.01	0.00	0.00	0.00
259	5.00	263	420	0	0.99	0.01	0.00	0.00	0.00
260	4.97	257	420	0	0.96	0.02	0.01	0.00	0.00
261	4.94	251	420	20	0.93	0.04	0.02	0.01	0.00
262	4.90	265	440	0	0.99	0.01	0.00	0.00	0.00
263	4.86	258	440	0	0.98	0.01	0.00	0.00	0.00
264	4.81	251	440	0	0.95	0.03	0.01	0.01	0.00
265	4.76	244	440	80	0.91	0.05	0.02	0.01	0.01
266	4.70	272	520	0	1.00	0.00	0.00	0.00	0.00
267	4.65	264	520	0	1.00	0.00	0.00	0.00	0.00
268	4.59	257	520	0	1.00	0.00	0.00	0.00	0.00
269	4.54	250	520	0	0.99	0.01	0.00	0.00	0.00
270	4.47	244	520	0	0.98	0.01	0.00	0.00	0.00
271	4.42	237	520	0	0.97	0.02	0.01	0.00	0.00
272	4.36	230	520	0	0.94	0.04	0.01	0.01	0.00
273	4.29	225	520	0	0.92	0.05	0.02	0.01	0.01
274	4.24	217	520	0	0.87	0.07	0.03	0.02	0.01
275	4.18	211	520	0	0.83	0.08	0.04	0.03	0.02
276	4.12	204	520	0	0.78	0.10	0.05	0.04	0.03
277	4.08	198	520	20	0.72	0.12	0.07	0.06	0.04
278	4.05	211	540	0	0.86	0.07	0.03	0.02	0.01
279	4.01	205	540	0	0.82	0.08	0.04	0.03	0.02
280	3.98	199	540	0	0.77	0.10	0.05	0.05	0.03
281	3.94	192	540	0	0.69	0.13	0.07	0.07	0.04
282	3.85	185	540	0	0.62	0.14	0.09	0.09	0.06
283	3.74	179	540	20	0.56	0.16	0.11	0.10	0.07
284	3.66	192	560	40	0.75	0.11	0.06	0.05	0.03
285	3.58	219	600	0	0.97	0.02	0.01	0.00	0.00
286	3.49	224	600	0	0.98	0.02	0.00	0.00	0.00
287	3.41	218	600	0	0.96	0.02	0.01	0.00	0.00
288	3.31	211	600	0	0.94	0.04	0.01	0.01	0.00
289	3.21	215	600	0	0.95	0.03	0.01	0.01	0.00
290	3.09	208	600	0	0.93	0.04	0.02	0.01	0.01
291	2.97	201	600	0	0.89	0.06	0.02	0.02	0.01
292	2.83	195	600	0	0.86	0.07	0.03	0.02	0.01
293	2.69	193	600	0	0.84	0.08	0.04	0.03	0.02
294	2.54	188	600	0	0.81	0.09	0.04	0.04	0.02
295	2.39	184	600	20	0.78	0.10	0.05	0.04	0.03
296	2.24	199	620	0	0.91	0.05	0.02	0.01	0.01
297	2.05	196	620	0	0.90	0.05	0.02	0.02	0.01
298	1.90	192	620	0	0.88	0.06	0.03	0.02	0.01
299	1.77	188	620	0	0.85	0.07	0.03	0.03	0.01
300	1.63	189	620	0	0.86	0.07	0.03	0.02	0.01
301	1.49	188	620	0	0.85	0.07	0.03	0.03	0.01
302	1.35	184	620	40	0.83	0.08	0.04	0.03	0.02
303	1.22	214	660	0	0.98	0.01	0.00	0.00	0.00
304	1.13	213	660	0	0.98	0.01	0.00	0.00	0.00
305	1.04	214	660	0	0.98	0.01	0.00	0.00	0.00
306	0.96	226	660	0	0.99	0.01	0.00	0.00	0.00
307	0.86	225	660	0	0.99	0.01	0.00	0.00	0.00
308	0.76	233	660	0	0.99	0.00	0.00	0.00	0.00
309	0.66	238	660	0	1.00	0.00	0.00	0.00	0.00
310	0.56	234	660	0	0.99	0.00	0.00	0.00	0.00
311	0.49	236	660	0	1.00	0.00	0.00	0.00	0.00
312	0.42	239	660	0	1.00	0.00	0.00	0.00	0.00
313	0.36	242	660	0	1.00	0.00	0.00	0.00	0.00
314	0.30	239	660	0	1.00	0.00	0.00	0.00	0.00
315	0.24	237	660	0	1.00	0.00	0.00	0.00	0.00
316	0.18	236	660	0	1.00	0.00	0.00	0.00	0.00
317	0.12	235	660	0	0.99	0.00	0.00	0.00	0.00
318	0.07	232	660	0	0.99	0.01	0.00	0.00	0.00
319	0.05	229	660	0	0.99	0.01	0.00	0.00	0.00
320	0.02	228	660	0	0.99	0.01	0.00	0.00	0.00
321	0.00	229	660	0	0.99	0.01	0.00	0.00	0.00
322	0.00	227	660	0	0.99	0.01	0.00	0.00	0.00