Proceedings of the 52nd Hawaii International Conference on System Sciences | 2019

Reinforcement learning for extended reality: designing self-play scenarios

Leonardo A. Espinosa Leal
Dept. of Business Management and Analytics
Arcada University of Applied Sciences and
Hanken School of Economics
Helsinki, Finland
leonardo.espinosaleal@arcada.fi

Anthony Chapman
Computing Department
University of Aberdeen
Aberdeen, UK
r01ac14@abdn.ac.uk

Magnus Westerlund
Dept. of Business Management and Analytics
Arcada University of Applied Sciences
Helsinki, Finland
magnus.westerlund@arcada.fi

Abstract

A common problem for deep reinforcement learning networks is a lack of training data to learn specific tasks through generalization. In this paper, we discuss using extended reality to train reinforcement learning agents to overcome this problem. We review popular reinforcement learning and extended reality techniques and then synthesize the information, this allowed us to develop our proposed design for a self learning agent. Meta learning offers an important way forward, but the agents ability to perform self-play is considered crucial for achieving successful AI. Therefore, we focus on improving self-play scenarios for teaching self-learning agents, by providing a supportive environment for improved agent-environment interaction.

1. Introduction

In this article, we propose a method and experimental study for a self-playing agent using deep reinforcement learning in an extended reality environment. Our aim is to address the opportunity to design self-play scenarios for self-learning agents that combine various sources for discovering and generating training data. By reviewing the literature we noticed a gap which could improve existing architectures used for creating self-learning agents.

Since 2012, the world has seen a rapid improvement in results from neural networks that use deep networks [1]. Some argue that when software exceeds what a human is capable of, artificial intelligence will make a leap into the future. An example of this can be, a deep network that can identify cancer from an image more efficiently than a doctor can [2]. Without further debating the philosophical definition of classical AI, as represented by, for example Asimov [3], and the view of deep learning aficionados, we consider that intelligence is often referred to as including the ability to automate tasks based on some type of reasoning. It may also constitute the ability to autonomously and continuously improve it's reasoning as a result of internal or external influences.

This type of reasoning is also gaining momentum in AI research in the wake of the success of DeepMind and their AlphaGo system that beat one of the best human Go players [4]. As we will discuss later, the ability of AlphaGo to perform probabilistic reasoning within its parameters and to, perhaps most importantly, continuously improve from both self-play and from human interaction has set the base line for so-called intelligent systems that can be considered achieving at least an elementary version of AI [5]. These are expert systems, they are very specialized and lack the ability to reason beyond their narrow ability. Deep reinforcement learning networks have improved the ability for models to generalize on complex task, but the exploration-vs-exploitation challenge is a difficult one to overcome. In [6] they suggest that an agent should focus on both guided meta learning, to improve the agents ability to perform the given tasks, and a self-play mechanism, which should improve the agents reasoning ability by testing different actions to take in a stochastic environment and giving preference to actions which give better results than others in such environments. Formulating an agents environment and goals is strenuous enough without the added complexities posed by also providing the agent with AI. The ability to combine both supervised interaction for meta learning and multi-agent self-play interaction (competitive or collaboration) may offer better results than using one without the other [5], [7].

For extended reality applications, we can envision agents (objects) with diverse behavior, interaction, and form that can make use of reinforcement learning (RL) as to improve their AI ability. Examples can be conjured in many domains, including education, gaming, and healthcare. In the following section we provide a review of various domains and relevant literature applying reinforcement learning agents in extended reality. Section 3 provides the conceptual foundations of reinforcement learning agents. In section

URI: https://hdl.handle.net/10125/59456
ISBN: 978-0-9981331-2-6
(CC BY-NC-ND 4.0)

4 we present a set of self-play scenarios that we will consider in our experiment. Section 5 presents an experimental design proposal. Finally, section 6 and 7 present the discussion and conclusion respectively .

2. Research methods and concepts

The paper can be divided into two discernible parts using complementary research methods. The first part is a literature review of the relevant state of important concepts. Through the literature review we provide a foundational understanding of root definitions for the second part where we provide an analysis of our findings. In this latter part we follow a soft systems methodology (SSM) by framing a problem formulation (model learning) and an action plan (conceptual model for learning) aimed at future research. Sørensen et al. [8] state that a SSM consists of an analysis of the current status of the system, including inherent problems and activities, a definition of said system deriving the actual goal of the targeted system (root definition) in order to propose a conceptual system model.

2.1. Extended reality

The term eXtended Reality (XR) has become known as a common term for fields were digitally enhanced environments and human-interaction are studied. This includes Virtual Reality (VR), Mixed Reality (MR), and Augmented Reality (AR). Compared to, for example, autonomous driving, these environments offer AI researchers an important and relatively low-cost setting for implementing human-interacting algorithms, such algorithms may contain reasoning that can be considered an avenue towards AI. Perhaps most importantly, these digital environments allow us to study an agents decision making, and thereby, offering a feed-back loop between the agent-environment-user.

Still, the need for training new abilities often require that agents are presented with big data. The traditional approach was often to gather that data from users, e.g. playing a game, and then train on this data. Today, the deep networks' need for massive data and relatively complex training scenarios for reinforcement learning presents researchers with a problem that is often better solved by augmenting additional data for training networks, than using real data. Data augmentation methods depend on the problem at hand, an agent may for example have to learn how to deal with object recognition, spatial actions to take in relation to detected objects, or temporal differences in scenarios, to name a few. The following sub-section reviews some of the relevant literature on how to perform data augmentation for training agents.

2.2. Data augmentation

In recent years, data augmentation has gained prominence as a studied method for extending available datasets [1]. The ability to train deep networks often depend on the availability of big data. The fields of both image recognition and voice recognition has been strongly influenced by deep learning methods, and this has motivated a focus on data augmentation. For RL many of the same concepts can be utilized, but there is also a need for methods that work particularly in the temporal dimension.

Traditional, naïve, approaches tend to manipulate the investigated environment or dataset in various ways. For visual tasks these have included scaling of objects, translating i.e. moving objects spatially to various positions, rotation of objects at various angles, flipping objects as to remove bias from any direction, adding noise, changing lightning conditions, and transforming perspective of a known object by changing the angle of view [1], [9].

For audio tasks, data augmentation often includes deformations into a temporal dimension. Approaches include time stretching by changing audio speed, pitch shifting by raising or lowering the tone frequency by various degrees, dynamic range compression, and introducing background noise both/either gaussian or natural noise [10].

The naïve data augmentation approaches tend to produce limited alternative data for RL agents to learn from in an extended reality setting. For an RL agent to learn new abilities, data augmentation must support the agents scenario learning process. As suggested by [11], we shift from learning to generalize on spatial data to reacting to continuous-time dynamical systems without a priori discretization of time, state, and action.

Several approaches exist for the creation of these scenarios. An important method is adversarial learning, as it can produce new and complex augmented datasets by pitting a generative model against an adversary [12]. A generative model in combination with XR, can also address the exploration problem, as exploring some states in the physical reality could be very costly and dangerous. This combination also allows the system developer to understand which state spaces in the virtual environment has been visited and trained upon, and the model's ability to generalize in the extended reality environment. These generative techniques for extending the learning environment are further explored in the following section.

3. Reinforcement learning

Although XR is slowly receiving more recognition, AI and machine learning's use of XR to enhance learning is lagging behind. XR could help improve an AI's behavior by providing information from either pure virtual or semi-real environments. Reinforcement learning is one machine learning technique which, when combined with XR, could produce interesting and beneficial results for many applications, such as driverless cars, autonomous factories, smart cities, gaming and more.

RL's primary purpose is to calculate the best action an agent should take when an environment is provided. With RL, we could be able to calculate what best action to take by maximizing the cumulative reward from previous actions, thus learning a policy. Although RL is still relatively young, it has received a lot of attention for its potential to advance and improve applications in gaming, robotics, natural language processing (including dialogue systems, machine translation, and text generation), computer vision, neural architecture design, business management, finance, healthcare, intelligent transportation systems, and other computer systems [13].

Attention and memory are two parts from RL which, if done impetuously, could negatively affect performance. Attention is the mechanism which focuses on the salient parts. Whereas, memory provides long term data storage, and attention is an approach for memory addressing [13]. Using XR and self-play, agents may be able to learn desired behavior before an action an agent makes become crucial to their performance. As an example, autonomous helicopter software could learn fundamental mechanisms for flight using virtual data in simulations in order to achieve high level of attention using the memory required, without the risks posed by real world applications. Once the attention has reached a desired level, it can be applied to real agents in the physical world.

General value functions can be used to represent knowledge. RL, arguably, mimics knowledge in the sense that it (generally) learns from the results of actions taken. Thus, one may be able to represent knowledge with general value functions using policies, termination functions, reward functions, and terminal reward functions as parameters [14]. Doing so, an agent may be able to predict the values of sensors, and policies to maximize those sensor values, and answer predictive or goal-oriented questions.

Generative Adversarial Networks (GANs) [12] estimate generative models via an adversarial process by training two models simultaneously, a generative model G to capture the data distribution, and a discriminative model D to estimate the probability that a sample comes from the training data but not the generative model G. Such an approach could be extended to XR by training a generative model G on virtual / simulated test data and then a discriminative model D to estimate the probability that a sample comes from the real world. This could help tackle some of the issues with RL within virtual environments and extended to the real world. RL and XR could be used before the agent is applied to a real environment, this could save on resources and make autonomous systems a more viable option for general use.

GANs together with transfer learning could advance self-play using virtual environments for real world agents [15]. By combining virtual data generative models and transferring the learning model to a discriminative model, we may be able to accurately express what was learned from the virtual learning environment to the real agent. Again, unforeseen problem will inevitably arise due to the nature of modeling. By using RL both in the virtual learning phase and embedded into the real agent, we may drastically improve a real agent's learning time.

Vezhnevets et al. [16] proposed strategic attentive writer (STRAW), a deep recurrent neural network architecture, for learning high-level temporally abstracted macro-actions in an end-to-end manner based on observations from the environment. Macro-actions are sequences of actions commonly occurring. STRAW builds a multi-step action plan, updated periodically based on observing rewards, and learns for how long to commit to the plan by following it without replanning. Similar to GANs, STRAW could be used after the simulation learning stage so the agent copes with any discrepancies between the simulation and the real world.

Adaptive learning is a core characteristic to achieving strong AI [17]. Several adaptive learning methods have been proposed which utilize prior knowledge [13, 18, 19]. [18] by representing a particular optimization algorithm as a policy, and convergence rate as reward. [13, 19] proposed to learn a flexible recurrent neural network (RNN) model to handle a family of RL tasks, to improve sample efficiency, learn new tasks in a few samples, and benefit from prior knowledge.

The notion of self-play is one of the biggest advancements of modern AI. AlphaGo AI is Deepmind's newest Go playing AI [20], that learns, *tabula rasa*, superhuman proficiency in challenging domains. Starting with the basic rules, they used self-play for the AI to learn strategies by playing against itself and storing efficient / rewarding moves.

Fictitious Self-Play, is a machine learning framework that implements fictitious play in a sample-based fashion [21].

The three strategies that are compared are: Learning by self-play, learning from playing against a fixed opponent, and learning from playing against a fixed opponent while learning from the opponents moves as well [5].

4. Self-play scenarios and architectures

It is very hard for an AI self-learning agent to generalize upon training into real scenarios. This problem is known as the *reality gap* [22]. In the initial stages of AI research, the training of self-learning agents included rules or limited scenarios where it learns and improves upon competition against other introduced players. Interestingly, videogames have emerged as one the main source of *benchmark* environments for the training and testing of such agents, mostly due to its realistic, yet controlled approach to the real world, and the easy access to large amounts of data. For instance, in a recent development, an AI agent is trained by playing with a perfect copy of itself without any supervision [23]. In this scenario, a set of basic rules of the game have been introduced at the beginning and the agent improves much faster using a vector of rewards instead of the *classical* scalar quantity [24].

Initially, self-play agents were trained to play boardgames (such as chess and go, among others) [20] but it has now been successfully extended from the classic and *simpler* Atari 2600 video games [25] to more complex first-person shooters (Doom [26], Battlefield [27], Quake [28]), Role Playing games[1](Warcraft [29]), Real-Time Strategic Games (Starcraft [30–32]) and more recently Multiplayer Online Battle Arena (Dota 2 [33]). For a more comprehensive review see the work by Justesen *et al* [34].

The motivation has increased in the recent years mainly due to the advances in neural network architectures suitable to the reinforcement learning paradigm: DQN [25], AC3 [35], DSR [36], Dueling networks [37] among others as well as the development of powerful and accessible GPU computing frameworks. This exciting area of research is broadly known as Deep Reinforcement Learning [13, 38, 39].

The challenge of training self-playing agents in order to develop more complex policies inside realistic and highly specific or general environments remain as an open problem. Most of the recent developments tend to focus on very particular properties of the learning agent

[1]Including the Massively Multiplayer Online type of games.

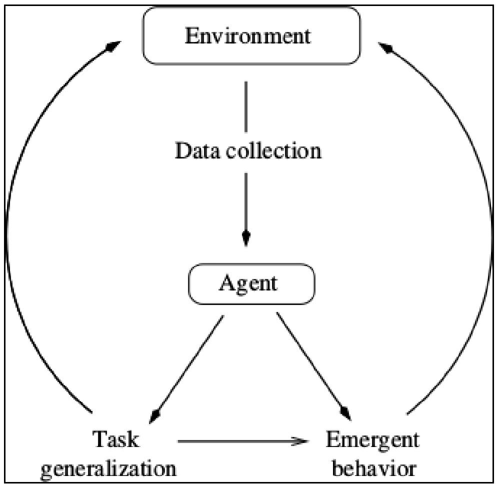

Figure 1. Proposed general architecture for a self-learning agent interacting with its environment.

or the way that they interact with their surroundings. To address this issue, we identify two general mechanism that can be improved in order to design a better self-learning agent: self-play scenarios and self-learning architectures.

4.1. Improving self-play scenarios and self-learning agents: closing the reality gap

Constructing realistic self-play scenarios plays a fundamental role in training self-learning agents. Once an agent is immersed in a specific environment, we expect (independently of the self-learning architecture) that it will learn a set of policies accordingly to the received experiences [2]. A problem which is widely understood, is that when agents learn from strict simulated scenarios, they may not be prepared for unexpected situations when the environment changes, such as a pigeon flying towards the sensor of a driverless car. Here, we propose a general scheme that uses the versatility of the videogames or simulators as a source of synthetic data and the wide array of capabilities of modern extended reality technologies, to enrich the properties of the real environment during the training of self-learning agents. The agent may learn independently, but the environment can be controlled to persuade the agent to learn a set of additional policies for unexpected scenarios. In addition to the enriched data, the self-learning agent may be trained using purely synthetic data. But the limitations of this method rely in

[2]A learning agent can learn a set of policies, or will optimize the parameters of a given set of policy. New policies can emerge even without previous knowledge. The sum of the whole policies is called general policy.

the accuracy of the representation of the real scenarios.

For the self-learning mechanism, we have identified three key steps which could improve the design of architectures for self-learning agents, this may improve policies both in terms of effectiveness and robustness. The three areas are: Data collection, task generalization and emergent behavior (see Fig. 1). For a given agent, in the first stage the agent will need to interact with the environment, possibly by accessing a data collection, then the agent should be able to generalize a set of given tasks and, simultaneously, new skills should emerge (independently or due to the task generalization). In the final stage, the emergent and the generalization skills interact with the environment to create a *continuous* self-learning agent.

To illustrate the steps, we present a set of representative developments in the area of (deep) reinforcement learning, shown in Figure 2. Each can be used as a building block inside a complete self-play scenario, for example, Figure 2a is a low-fidelity rendered images with random camera positions, lighting conditions, object positions, and non-realistic textures use to train a self-learning agent (from Ref [40]), Figure 2b is an agent which uses a compressed representation of a real scenario to learn a set of policies which are successfully use back to the real environment. (from Ref [41]) Figure 2c shows an image of a robot used as a one-demonstration example during the training stage. (from [42]), Figure 2d shows another image of a robot used during a training stage to teach a robot to place an object in a target position.(from [43]), Figure 2e depicts an agent stacking two blocks, behavior learn from sparse rewards (from [7]), Figure 2f illustrates one competitive environment where one the agents develops a new set skills (from Ref [44]).

which can be used as building blocks inside a complete self-play scenario (see Fig 2).

1. Data collection:
   Domain randomization (DR): in a recent communication by one of the DeepMind research teams, it was showed that an agent can be trained on artificially generated scenarios. In this paper, the authors successfully transfer the knowledge from a neural network purely trained on low resolution rendered RGB images: *domain randomization* [40]. This method can be extensively use for training agents in the case that the amount of data available is low or when the separation between the real and the train environment is immense.

   World models (WM): a recent communication by Ha et *al.* [41] showed that self-learning agents can be trained in a compressed spatial and temporal representation of the environment. This method is highly powerful because the agent can learn in a more compact or *hallucinated* universe and then go back to the original environment exporting the set of learned abilities. One of the main advantages of this method is the possibility to perform a much faster and accurate *in situ* training of the agents by using less demanding computational resources.

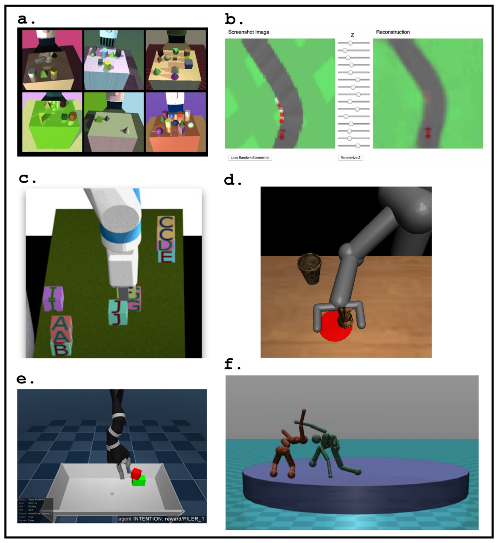

Figure 2. Illustration of how different components from Figure 1 can be used on complete self-learning systems.

2. Task generalization:

   One-shot imitation learning (OSIL): the authors present an improved method that uses a meta-learning framework built upon the soft attention model [45] named *one-shot imitation learning* [42]. Here, the agents are able to learn a new policy and solve a task after being exposed to a few demonstrations during the training stage.

   One-shot visual imitation learning (OSVIL): Meta-imitation learning algorithm that teaches an agent to learn how to learn efficiently [43]. In this work, a robot reuses past experiences and then upon a single demonstrations, it develops new skills.

3. Emergent behavior:

   Scheduled auxiliary control (SAC): another research team from DeepMind introduced a new framework that allows agents to learn new and

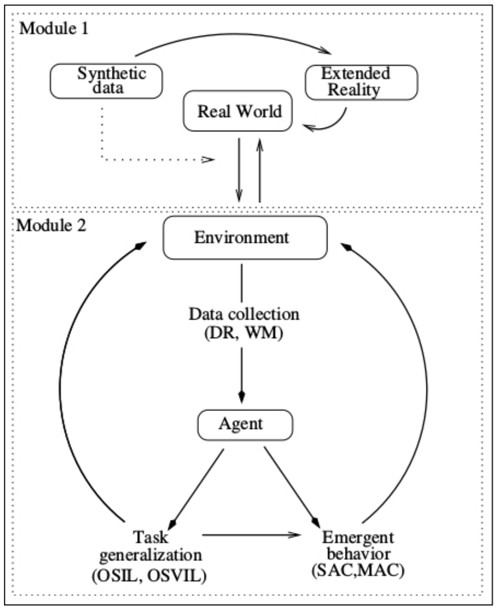

Figure 3. Two module design of a general architecture for a self-learner agent interacting with its enriched or altered environment.

complex behaviors in presence of a stream of sparse rewards [7].

Multi-agent competition (MAC): a paper by one research team from OpenAI, the authors showed that multi-agents self-play on complex environments can produce behaviors which can be more complex than the environment itself [44]. The emergent skills can improve the capabilities of the agents upon unexpected changes in the real environment.

To summarize, Figure 1 illustrates how an agent can retrieves data from an environment and then generalizes to a specific task and simultaneously develops new abilities. The new skills can emerge independently or due to the task generalization process. In the final stage, the environment gets modified by the agent itself. A combination of such methods could be used to create more effective architectures for teaching self-learning agents. In the next section we present a general design and then we discuss the possible applications concerning to specific cases.

5. Proposed Design

As already discussed, our goal of designing general self-play scenarios for teaching self-learning agents can be tackled by separating the data retrieved from the environment and the agent's self-learning architecture. In the spirit of the SMM (see Section 2) we present a general scheme in which we divide the general architecture into two modules, in Module 1, the agent retrieves the data from its surroundings as a combination of information from the real world and synthetic data (or pure synthetic data), and in Module 2 (equivalent to the structure of the Fig 2), the agent creates its final policies. The general scheme is depicted in the Fig 3,

For a self-learning agent inside a specific self-learning scenario, there is not a difference between the synthetic or the real data. Here we call real data the information extracted from physical world without any previous or further digital modifications. The agent uses exclusively the information, in terms of raw bytes, independently of the sensors that connect it with the environment. The use of synthetic data arises as a need to expose the self-learning agent to unexpected situations or conditions that allow it to create a set of related policies. It can be also necessary to increase the amount of available training data (data augmentation).

5.1. Applications

The aforementioned improvements can be employed in specific applications using specific designs. In particular, by using the new and open simulation frameworks such as OpenAI Gym[3] or Dopamine[4] among others. Here we discuss two particular cases, however the ideas can be exported to other possible applications as well.

- Self-driving machines: One of the most important applications of self-learning agents is the self-driving machines, in particular self-driving cars [46, 47]. The available datasets for training are increasing and recently, the research community is exploring the use of a combination of synthetic and real data [48]. The particular design of the self-learning module, in this case, must encourage the creation of emergent behavior policies, however the discussion about the imposition of the security of the passengers above the pedestrians is still a open philosophical question. One important characteristic is that the architecture should carefully avoid the modification of the environment, even upon emergent behaviors. The use of simulated data similar to modern video games engines can be of paramount importance for the creation of optimal policies.
- Robotic surgery: One of the most discussed

[3]https://github.com/openai/gym
[4]https://github.com/google/dopamine

applications, in particular due to the ethical implications [49]. In this case the modifications of the environment is unavoidable. However, the architecture must to control the submodule for emergent behaviors, in particular in the case of highly sensitive interventions. The agent should be trained to accomplish a large set of general tasks.

6. Discussion & Limitations

The design of self-learning and self-play scenarios is still an area of fruitful developments and research. Many critics have pointed out that AI research is limited by its own ideas [50]. The creation and discussion of general architectures can open the door to new proposals, in particular, for the final emergence of the expected Artificial General Intelligence (AGI) agent. Despite a boom in the field during the last years, there are many open questions about how to enable self-learning agents to achieve specific tasks without any supervision. We propose to tackle this question by using our general architecture, which can be, in principle, limited by the modular developments and the availability of specific dataset or sensors.

7. Conclusion & Future Work

In this work we presented a general review and we designed a general architecture for self-learning agents. Our design included two separate modules, one for the creation of the data and the second for the independent self-learning stage. We conclude, that the second module is, in general, divided into three stages where in principle, each one only is in charge of accomplishing an independent task: data collection, task generalization and emergent behavior. In very particular designs, generalization can influence emergent behaviors, but only in one direction. We are currently working on implementing our proposed designed and plan to publish our findings in the near future.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[2] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun, "Dermatologist-level classification of skin cancer with deep neural networks," *Nature*, vol. 542, no. 7639, p. 115, 2017.

[3] I. Asimov, "Runaround. I, Robot," *New York: Bantam Dell*, 1950.

[4] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, *et al.*, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," *arXiv preprint arXiv:1712.01815*, 2017.

[5] M. Van Der Ree and M. Wiering, "Reinforcement learning in the game of othello: learning against a fixed opponent and learning from self-play," in *Adaptive Dynamic Programming And Reinforcement Learning (ADPRL), 2013 IEEE Symposium on*, pp. 108–115, IEEE, 2013.

[6] T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, B. Yang, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, *et al.*, "Never-ending learning," *Communications of the ACM*, vol. 61, no. 5, pp. 103–115, 2018.

[7] M. Riedmiller, R. Hafner, T. Lampe, M. Neunert, J. Degrave, T. Van de Wiele, V. Mnih, N. Heess, and J. T. Springenberg, "Learning by playing-solving sparse reward tasks from scratch," *arXiv preprint arXiv:1802.10567*, 2018.

[8] C. Sørensen, S. Fountas, E. Nash, L. Pesonen, D. Bochtis, S. M. Pedersen, B. Basso, and S. Blackmore, "Conceptual model of a future farm management information system," *Computers and electronics in agriculture*, vol. 72, no. 1, pp. 37–47, 2010.

[9] P. Pai, "Data augmentation techniques in CNN using Tensorflow." `https://bit.ly/2KLm8K6`, 2017. Accessed: 2018-06-12.

[10] J. Salamon and J. P. Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 279–283, 2017.

[11] K. Doya, "Reinforcement learning in continuous time and space," *Neural computation*, vol. 12, no. 1, pp. 219–245, 2000.

[12] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[13] Y. Li, "Deep reinforcement learning: An overview," *arXiv preprint arXiv:1701.07274*, 2017.

[14] R. S. Sutton, J. Modayil, M. Delp, T. Degris, P. M. Pilarski, A. White, and D. Precup, "Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction," in *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pp. 761–768, International Foundation for Autonomous Agents and Multiagent Systems, 2011.

[15] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[16] A. Vezhnevets, V. Mnih, S. Osindero, A. Graves, O. Vinyals, J. Agapiou, *et al.*, "Strategic attentive writer for learning macro-actions," in *Advances in neural information processing systems*, pp. 3486–3494, 2016.

[17] B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman, "Building machines that learn and think like people," *Behavioral and Brain Sciences*, vol. 40, 2017.

[18] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[19] Y. Chen, M. W. Hoffman, S. G. Colmenarejo, M. Denil, T. P. Lillicrap, M. Botvinick, and N. de Freitas, "Learning to learn without gradient descent by gradient descent," *arXiv preprint arXiv:1611.03824*, 2016.

[20] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, *et al.*, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," *arXiv preprint arXiv:1712.01815*, 2017.

[21] J. Heinrich, M. Lanctot, and D. Silver, "Fictitious self-play in extensive-form games," in *International Conference on Machine Learning*, pp. 805–813, 2015.

[22] N. Jakobi, P. Husbands, and I. Harvey, "Noise and the reality gap: The use of simulation in evolutionary robotics," in *European Conference on Artificial Life*, pp. 704–720, Springer, 1995.

[23] A. Dosovitskiy and V. Koltun, "Learning to act by predicting the future," *arXiv preprint arXiv:1611.01779*, 2016.

[24] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction," 2011.

[25] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[26] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaśkowski, "Vizdoom: A doom-based ai research platform for visual reinforcement learning," in *Computational Intelligence and Games (CIG), 2016 IEEE Conference on*, pp. 1–8, IEEE, 2016.

[27] J. Harmer, L. Gisslén, H. Holst, J. Bergdahl, T. Olsson, K. Sjöö, and M. Nordin, "Imitation learning with concurrent actions in 3d games," *arXiv preprint arXiv:1803.05402*, 2018.

[28] C. Beattie, J. Z. Leibo, D. Teplyashin, T. Ward, M. Wainwright, H. Küttler, A. Lefrancq, S. Green, V. Valdés, A. Sadik, *et al.*, "Deepmind lab," *arXiv preprint arXiv:1612.03801*, 2016.

[29] P.-A. Andersen, M. Goodwin, and O.-C. Granmo, "Towards a deep reinforcement learning approach for tower line wars," in *International Conference on Innovative Techniques and Applications of Artificial Intelligence*, pp. 101–114, Springer, 2017.

[30] O. Vinyals, T. Ewalds, S. Bartunov, P. Georgiev, A. S. Vezhnevets, M. Yeo, A. Makhzani, H. Küttler, J. Agapiou, J. Schrittwieser, *et al.*, "Starcraft ii: a new challenge for reinforcement learning," *arXiv preprint arXiv:1708.04782*, 2017.

[31] G. Synnaeve, N. Nardelli, A. Auvolat, S. Chintala, T. Lacroix, Z. Lin, F. Richoux, and N. Usunier, "Torchcraft: a library for machine learning research on real-time strategy games," *arXiv preprint arXiv:1611.00625*, 2016.

[32] Y. Tian, Q. Gong, W. Shang, Y. Wu, and C. L. Zitnick, "Elf: An extensive, lightweight and flexible research platform for real-time strategy games," in *Advances in Neural Information Processing Systems*, pp. 2656–2666, 2017.

[33] "Dota 2." `https://blog.openai.com/dota-2/`. Accessed: 2018-06-12.

[34] N. Justesen, P. Bontrager, J. Togelius, and S. Risi, "Deep learning for video game playing," *arXiv preprint arXiv:1708.07902*, 2017.

[35] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International Conference on Machine Learning*, pp. 1928–1937, 2016.

[36] T. D. Kulkarni, A. Saeedi, S. Gautam, and S. J. Gershman, "Deep successor reinforcement learning," *arXiv preprint arXiv:1606.02396*, 2016.

[37] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. De Freitas, "Dueling network architectures for deep reinforcement learning," *arXiv preprint arXiv:1511.06581*, 2015.

[38] S. S. Mousavi, M. Schukat, and E. Howley, "Deep reinforcement learning: an overview," in *Proceedings of SAI Intelligent Systems Conference*, pp. 426–440, Springer, 2016.

[39] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "A brief survey of deep reinforcement learning," *arXiv preprint arXiv:1708.05866*, 2017.

[40] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pp. 23–30, IEEE, 2017.

[41] D. Ha and J. Schmidhuber, "World models," *CoRR*, vol. abs/1803.10122, 2018.

[42] Y. Duan, M. Andrychowicz, B. Stadie, O. J. Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba, "One-shot imitation learning," in *Advances in neural information processing systems*, pp. 1087–1098, 2017.

[43] C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine, "One-shot visual imitation learning via meta-learning," *arXiv preprint arXiv:1709.04905*, 2017.

[44] T. Bansal, J. Pachocki, S. Sidor, I. Sutskever, and I. Mordatch, "Emergent complexity via multi-agent competition," *arXiv preprint arXiv:1710.03748*, 2017.

[45] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[46] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[47] N. J. Goodall, "Machine ethics and automated vehicles," in *Road vehicle automation*, pp. 93–102, Springer, 2014.

[48] H. A. Alhaija, S. K. Mustikovela, L. Mescheder, A. Geiger, and C. Rother, "Augmented reality meets computer vision: Efficient data generation for urban driving scenes," *arXiv preprint arXiv:1708.01566*, 2017.

[49] A. Wedmid, E. Llukani, and D. I. Lee, "Future perspectives in robotic surgery," *BJU international*, vol. 108, no. 6b, pp. 1028–1036, 2011.

[50] J. Pearl and D. Mackenzie, *The Book of Why: The New Science of Cause and Effect*. Basic Books, 2018.