Button Mashing: On the effectiveness of biased random inputs in reinforcement learning environments for Mario

Ferdinand Mathia, *Member, Univerity of applied Research Augsburg*

***Abstract*—The state-of-the-art deep reinforcement learning methods take hours or days to learn how to beat even the first level of *Super Mario Bros.*, but the best solutions for beating *Super Mario Bros.* that human can find consist of mostly a few actions. With this knowledge we hand made a biased random Agent that beats the first level in 0.5% of the time. Random input beats the level in fewer steps than the state of the art RL algorithm, but once the RL agent is trained it is a lot more consistent.**

I. Introduction

IT is common praxis to evaluate deep reinforcement learning (RL) agents in relative to the performance of an agent that presses random buttons [1]. A totally random Policy like this, makes very little progress in even the first level of Super Mario Bros [2]. If you compare the distribution of inputs of random agent with that of human experts, you recognise that, the distribution of button combinations is heavily biased toward a few of the 255 possible combinations (see section III). If human input is this heavily biased, it raises the question if it makes sense to use totally random sampling for the baseline random agent.

To answer this question we present our state-of-the-art deep reinforcement learning agent, analyse human input distribution and evaluate a biased and unbiased random agent.

II. State of the art

A. Reinforcement Learning

The agent interacts with an environment at time-step t. To-do this the agent selects an action A_t and it receives a reward R_t which signifies the quality of the chosen Actions, additionally it gets an Observation O_{t+1}. IT then uses this to chooses the action for the next step A_{t+1}. All the Environments used here have finite length and discrete action-spaces. [3]

The highest goal of RL is to find an optimal policy, the one that achieves the highest possible reward for a given environment. Depending on the environment there can be one or many optimal policies. Changing the environment may not lead to change in the optimal policies. Changing the enviroment without changing the policies is called reward shaping. This has been used to make environment easier to learn without affecting the quality of the result. [4]

We focused on Q-Learning agents popularized by Mnih, Kavukcuoglu, Silver, *et al.* [1], but there are many more formulations for agents. One type that should be mentioned are *Monte Carlo Methods*, because they will come up as a reasonable alternative in Section VIII. In Monte Carlo Policy Evaluation we take a completed episode and assign a value to each state covered by taking the discounted reward sum as the value of the state value function. We then combine the state value functions of multiple rollouts/ episodes by averaging the value of each state over the multiple episodes. Monte Carlo Methods can be applied in an Off-Policy context, that means that it policy π_1 can be optimized with episodes made with another policy π_2. Q-Learning is generally done On-Policy, were the π_1 can only be trained with experiences (O_t, A_t, R_t) made with π_1.

B. Rainbow RL

The RAINBOW agent [5] is a Q-Learning algorithm at heart, it combines the initial Deep Q-Network agent of Mnih, Kavukcuoglu, Silver, *et al.* [1] with the multitude of improvements made since its inception. The techniques include but are not limited to prioritized experience replay, noisy nets, lessons learned from double- and duelling deep Q-networks and "Learning from multi-step boot- strap targets"[5]. These additions to the network added complexity, but it is state-of-the-art when it comes to playing the set 52 Atari games, that are used to evaluate the ability of deep RL agents.

The Agent used in the following work is an open source reproduction [6] of the rainbow agent from [5] which in turn shares a lot of similarities with [1].

C. Open Ai Retro Contest

Current work in Reinforcement Learning Research is expanding from Atari games to newer and more graphically complex games. One of these efforts is the Open Ai Retro Contest. It focuses on transfer and meta learning between many Sega Genesis games [7]. But it also offers a unified interface and RL environment specification (e.g. possible actions and reward function) for many games on different consoles. Research into these arguably more complicated games like Super Mario has been splintered until now [2]. This large scale unification effort has the power to change this. This made it part of the motivation to provide a baseline of the start of the art in the standardized environment, because Mario has been an example that can draw public attention [8].

III. Action Distribution

We considered to multiple sources of human made inputs. First the play of the Authors, admittedly these can just barley

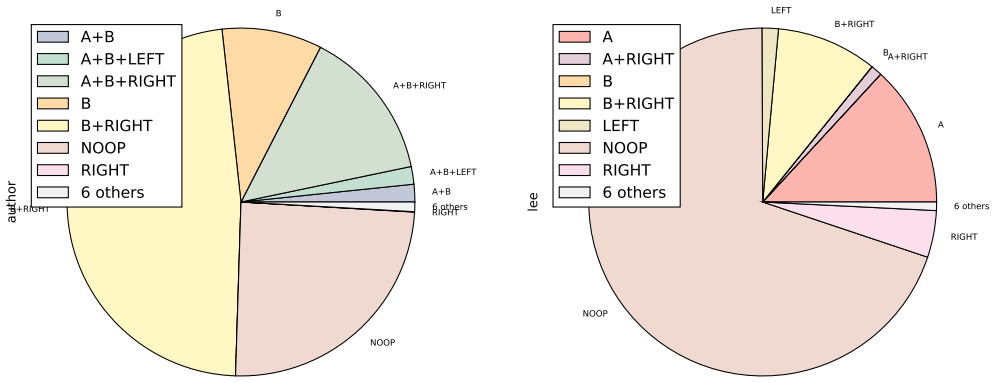

Fig. 1. Distribution of different button combinations. Left is the human play for beating world 1-1 made by the authors. Right best human made solution for beating the game as fast as possible (any% warpless see, Section III)

be described as human experts. The second source where "Tool Assisted Speedruns" (TAS), these are sets of inputs produced by human experts (experts in playing the specific game). TASs are produced by playing and replaying the game frame by frame and selecting the optimal action. They take hours of real-time to produce a couple of minutes of game time [9]. They represent the best solution, for beating games as fast as possible, that humans have been able to find. Tasvideos.org is a community driven website that lists and ranks these so called *movies*. The two movies used here were the "warpless any percent" [10] and "warped any percent" [11] by Happylee (pseudonym) they are the fastest in their respective category.

Based on the observation, that the movie of the play of the author contained mostly the two combinations of B + Right (47%) and A + B + Right (14%) we selected these two combinations as the only valid inputs of our biased random agent (see also Figure 1). With informal experimentation we settled on weighting the 3 buttons to 90% of the time and the 2 to 10%.

The unbiased agent chooses uniformly randomly between all the 11 actions available to the rainbow agent (specification in Section V).

IV. Reward function and evaluation metrics

The speedruning scene measures the level/ quality of play by how fast a certain end state can be reached, this differs form reward functions commonly used for RL and Mario [2]. The reason for the differing metrics is that the completion time is only available after completing a defined segment of play, which makes it an exceptional bad reward function. Delayed rewards present a large challenge for RL algorithms [3], and the end of an episode is the most delayed a reward can be. Because the metrics used by speedrunners are ill fitted, the reward in retro environment is defined as progress in the level in pixels scrolled since the start of the level.

We use completion as a metric for evaluating the different agents, We take into consideration that the agent does not optimize directly for this metric, which on some level can influence the quality of the metric itself. For a concrete example of reward function and evaluation metrics differing and the potential harmful effects see Section 2.1.4 of Leike, Martic, Krakovna, *et al.* [12].

V. Experimental Setup

- 4×10^6 Steps
- Training took 9 hours wall clock time, on two 16 core Intel Xeon processors with 32 GB ram and one Nvidia Titan X graphics card
- Other hyper-parameters were taken from [6], [5] and [1].
- The rainbow agent was trained on the "SuperMarioBros-Nes" environment in the "World1-1" scenario.
- The observation of a single step contains the last four frames.
- The action-space space differs from the default space of Retro. It contains only 11 actions (retro allows 37). The available actions are A , B, LEFT, RIGHT, DOWN and the combinations of A or B and one of the directions (e.g. A + LEFT).

For the rainbow agent we defined am episode as a single live in Mario but consecutive episodes where in the same game of Mario. A game in Mario is the time between pressing start on the title screen and the game over screen. This is also known as *Episodic Life* a method popularized by Mnih, Kavukcuoglu, Silver, *et al.* [1] and was used while training the Rainbow agent. For the Evaluation of the trained agent and for the random agents we defined an episode as only a single life in Mario and resetting to the beginning of 1-1 with full lives. We refer to this as *single life*. This makes the total reward easier to interpret, because there is no chance that the agent spawns at the checkpoint, which makes the distance to the end of the level constant, which in turn makes the reward of getting to the end constant.

For the rainbow the action of a single step gets repeated for three frames. For the random agent the actions get repeated four times. These values are taken form the works [6] and [1] respectively.

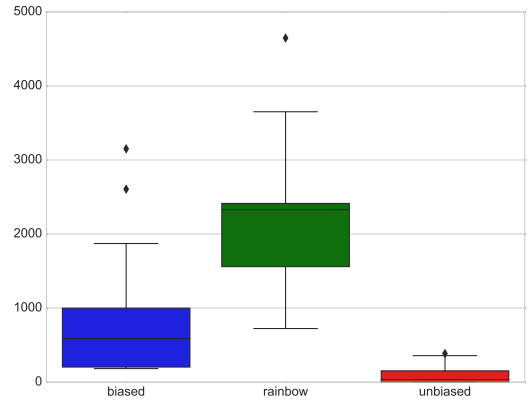

Fig. 2. The distribution of total rewards per episode in Super Mario Bros. The unbiased agent (red) never accomplished more than 1000 reward, with most episodes ending at or before the first enemy. The biased one (blue) has higher median value and outliers around 3200 which is the end of the level. The trained rainbow agent (green) has distribution skewed far higher, but this was accomplished through hours of compute. The values higher 3200 than are rewards gathered in the next world 2-1

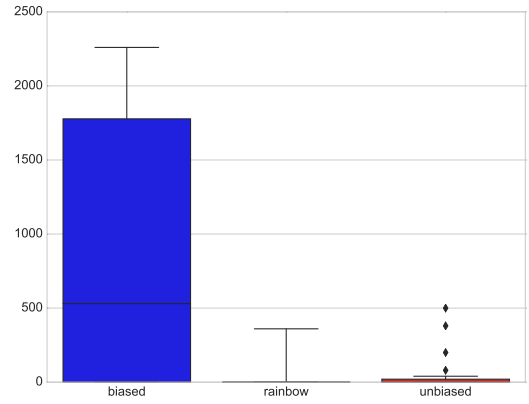

Fig. 3. The total reward per episode in the second game Super Mario World. The unbiased and biased random agents where unchanged other than changing which buttons they press, because Mario World uses X and A for running and jumping. For the trained agent the same transformation was done. It does poorly compared to the other agents. The 1.5 IQR whisker of Rainbow is higher than that of the unbiased agent, but the other descriptive statistics of the distribution are worst for the transferred rainbow agent.

VI. Strength of Random Agent

The random agent with an unbiased sampling of inputs barely makes it past the first Goomba (total reward $\sim$ 200) and never came close to finishing the level (sample size 150 episodes (one live per episode)). The biased agent is a lot better it receives around 500 reward per single live most frequently. With a sample size of 200 episodes it beat level 1-1 one time and died close to the goal once. The biased agent was able to achieve this in 20 Minutes on a Laptop with only an Intel i3 processor. The RL agents beats the level after 611 episodes, 350,000 steps or 8 and half hours wall-clock time, on a two Xeon CPU machine.

VII. Strength of trained Agent

The clear benefit of the trained Agent is that it is more consistent in playing good episodes than biased random searches. It beat the first level in 6 out of 32 single live episodes. In comparison, the random agent only beat it in 1 out of 200.

The trained agent also has a higher theoretical ceiling, if we evaluate it with the "time to completion" metric, because the expert human play clearly uses the left arrow button (see Figure 1). Examples where different inputs are important include moving left into the warp zones, standing still while waiting for moving platforms or collection the moving power ups. It is simply not possible for the biased random agent to input these buttons, but the used reward function does not reflect this negativity of our higher order evaluation function.

One of the benefits of the Open AI retro suite, is that we can see the generalization of the agents to other visually and gameplay similar environments [7]. Short inspection showed that the agent does generalizes to other visually similar environments about as wall as the unbiased random agent. The agent trained for Super Mario Bros. World 1-1 was transferred to World 2-1 and World 3-1 of the same game and to the different game "Super Mario World" level "Doughnut Hills 1".

VIII. Outlook

An obvious way to close the difference between biased random and trained agent is to improve the learning algorithm. A possible way to do that are to include the expert human play as experience in the agent [13]. Another way is re-framing the problem as episodic learning, to get an agent that learns a lot quicker but at the cost of some quality of the policy [14] [15].

The quality of the reward function also has huge impact on the learning rate [2] [3] . In the formulation we use this is seen, as changing the environment but this should not deter one form working on this aspect. There are theoretical methods to change the characteristic of the reward without changing the optimal policy [16]. This can be accomplished by human experts [17] [2], done with simple/ amateur human feedback [18] or fully automatically [19].

Apart from these more generic improvements this work raises the question of how effective a combination of random inputs with Monte Carlo search with simple rolouts can be.

Taking the input distribution from human play represents a simple way to integrate human experience. You could try collecting positive random experiences to prime the learning agent with. This experience could then be feed into the experience replay [13] or used to pre-train the controlling neural network.

IX. Conclusion

The Fact that a *random* policy was at least able to beat Super Mario Bros. further reinforces the question if current deep reinforcement learning is appropriate method for dealing with simpler environments like some video games. There are certainly more challenges in other games like "Montezuma's Revenge" [13]. Simple search algorithms also perform surprisingly well with Mario [20] [21]. This leads to the question if these complicated policies could be evaluated more appropri-

ately on other environments like 3D games [22] or complicated boardgames [23].

Further more the common practice of using unbiased random policies as baselines could be re-examined. If a biased random policy performs significantly better than unbiased, the *percentage of human performance* metric could be made more meaningful for these *easier* games.

APPENDIX
SUPPLEMENTARY MATERIALS

The source code for this project is available at https://r-n-d.informatik.hs-augsburg.de:8080/ferdinandpeer.mathia/starman. Videos and raw statistics of the training and evaluation of the various agents can be found at http://www.hs-augsburg.de/homes/mathiaf/data-science/.

REFERENCES

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, and others, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[2] F. P. Mathia, *Evaluation von deep reinforcement learning methoden anhand von „super mario bros.“* de, 2017. [Online]. Available: http://www.hs-augsburg.de/~mathiaf/ba/Thesis_Ferdinand_Mathia_2017.pdf (visited on Jun. 12, 2018).

[3] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, English. Cambridge, Mass.: MIT Press, 1998, ISBN: 978-0-262-19398-6. [Online]. Available: http://ieeexplore.ieee.org/servlet/opac?bknumber=6267343 (visited on Nov. 21, 2016).

[4] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *ICML*, vol. 99, 1999, pp. 278–287. [Online]. Available: http://www.cdf.toronto.edu/~csc2542h/fall/material/csc2542f16_reward_shaping.pdf (visited on Feb. 1, 2017).

[5] M. Hessel, J. Modayil, H. van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. G. Azar, and D. Silver, "Rainbow: Combining Improvements in Deep Reinforcement Learning," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*, S. A. McIlraith and K. Q. Weinberger, Eds., AAAI Press, 2018. [Online]. Available: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17204 (visited on Jun. 11, 2018).

[6] A. Nichol, *Anyrl-py: A reinforcement learning framework*, Jun. 2018. [Online]. Available: https://github.com/unixpickle/anyrl-py (visited on Jun. 12, 2018).

[7] A. Nichol, V. Pfau, C. Hesse, O. Klimov, and J. Schulman, "Gotta Learn Fast: A New Benchmark for Generalization in RL," *ArXiv:1804.03720 [cs, stat]*, Apr. 2018. arXiv: 1804.03720 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1804.03720 (visited on Jun. 12, 2018).

[8] J. Pearson, *Why Artificial Intelligence Researchers Love 'Super Mario Bros.'* Zugriff: 08.03.17, Oct. 2015. [Online]. Available: https://motherboard.vice.com/en_us/article/why-artificial-intelligence-researchers-love-super-mario-bros (visited on Mar. 8, 2017).

[9] A. Cecil, I. Liusvaara, and Jordan Potter, "Poémon Plays Twitch," en, *PoC —— GTFO 0x10*, p. 88, Jan. 2016.

[10] L. Tianda and Y. Yuli, *TASVideos submissions: #5975: HappyLee & Mars608's NES Super Mario Bros. "warpless" in 18:36.78*, May 2018. [Online]. Available: http://tasvideos.org/5975S.html (visited on Jun. 28, 2018).

[11] L. Tianda, *TASVideos submissions: #2964: HappyLee's NES Super Mario Bros "warped" in 04:57.31*, Dec. 2010. [Online]. Available: http://tasvideos.org/2964S.html (visited on Jun. 28, 2018).

[12] J. Leike, M. Martic, V. Krakovna, P. A. Ortega, T. Everitt, A. Lefrancq, L. Orseau, and S. Legg, "AI Safety Gridworlds," *ArXiv:1711.09883 [cs]*, Nov. 2017. arXiv: 1711.09883 [cs]. [Online]. Available: http://arxiv.org/abs/1711.09883 (visited on Jun. 12, 2018).

[13] T. Pohlen, B. Piot, T. Hester, M. G. Azar, D. Horgan, D. Budden, G. Barth-Maron, H. van Hasselt, J. Quan, M. Večerík, M. Hessel, R. Munos, and O. Pietquin, "Observe and Look Further: Achieving Consistent Performance on Atari," *ArXiv:1805.11593 [cs, stat]*, May 2018. arXiv: 1805.11593 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1805.11593 (visited on Jun. 22, 2018).

[14] C. Blundell, B. Uria, A. Pritzel, Y. Li, A. Ruderman, J. Z. Leibo, J. Rae, D. Wierstra, and D. Hassabis, "Model-Free Episodic Control," *ArXiv:1606.04460 [cs, q-bio, stat]*, Jun. 2016. arXiv: 1606.04460 [cs, q-bio, stat]. [Online]. Available: http://arxiv.org/abs/1606.04460 (visited on Mar. 25, 2017).

[15] A. Pritzel, B. Uria, S. Srinivasan, A. Puigdomènech, O. Vinyals, D. Hassabis, D. Wierstra, and C. Blundell, "Neural Episodic Control," *ArXiv:1703.01988 [cs, stat]*, Mar. 2017. arXiv: 1703.01988 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1703.01988 (visited on Mar. 7, 2017).

[16] E. Wiewiora, "Potential-based shaping and Q-value initialization are equivalent," *J. Artif. Intell. Res.(JAIR)*, vol. 19, pp. 205–208, 2003. [Online]. Available: http://www.aaai.org/Papers/JAIR/Vol19/JAIR-1907.pdf (visited on Dec. 13, 2016).

[17] A. Harutyunyan, S. Devlin, P. Vrancx, and A. Nowé, "Expressing Arbitrary Reward Functions as Potential-Based Advice.," in *AAAI*, 2015, pp. 2652–2658. [Online]. Available: http://anna.harutyunyan.net/wp-content/papercite-data/papers/harutyunyan2015aaai.pdf (visited on Feb. 9, 2017).

[18] A. Harutyunyan, T. Brys, P. Vrancx, and A. Nowé, "Shaping mario with human advice," in *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, International Foundation for Autonomous Agents and Multiagent Systems, 2015, pp. 1913–1914. [Online]. Available: http://dl.

acm.org/citation.cfm?id=2773501 (visited on Dec. 13, 2016).

[19] J. A. Arjona-Medina, M. Gillhofer, M. Widrich, T. Unterthiner, and S. Hochreiter, "RUDDER: Return Decomposition for Delayed Rewards," *ArXiv:1806.07857 [cs, math, stat]*, Jun. 2018. arXiv: 1806.07857 [cs, math, stat]. [Online]. Available: http://arxiv.org/abs/1806.07857 (visited on Jun. 22, 2018).

[20] Tom Murphy VII, "The first level of Super Mario Bros. is easy with lexicographic orderings and time travel," *The Association for Computational Heresy (SIGBOVIK) 2013*, 2013.

[21] S. Karakovskiy and J. Togelius, "The mario ai benchmark and competitions," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 1, pp. 55–67, 2012. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6156425 (visited on Dec. 1, 2016).

[22] M. Jaderberg, W. M. Czarnecki, I. Dunning, L. Marris, G. Lever, A. G. Castaneda, C. Beattie, N. C. Rabinowitz, A. S. Morcos, A. Ruderman, N. Sonnerat, T. Green, L. Deason, J. Z. Leibo, D. Silver, D. Hassabis, K. Kavukcuoglu, and T. Graepel, "Human-level performance in first-person multiplayer games with population-based deep reinforcement learning," *ArXiv:1807.01281 [cs, stat]*, Jul. 2018. arXiv: 1807.01281 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1807.01281 (visited on Jul. 9, 2018).

[23] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of Go with deep neural networks and tree search," en, *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016, ISSN: 0028-0836. DOI: 10.1038/nature16961. [Online]. Available: http://www.nature.com/nature/journal/v529/n7587/full/nature16961.html (visited on Nov. 15, 2016).