AI

MDPI

Article

Refined Continuous Control of DDPG Actors via Parametrised Activation

Mohammed Hossny [1,*,†], Julie Iskander [2,†], Mohamed Attia [3,†], Khaled Saleh [4,†] and Ahmed Abobakr [5,†]

1 School of Engineering and IT, University of New South Wales, Canberra, ACT 2612, Australia
2 Walter and Eliza Hall Institute of Medical Research, Melbourne, VIC 3052, Australia; iskander.j@wehi.edu.au
3 Medical Research Institute, Alexandria University, Alexandria 21568, Egypt; mohamed.hassan.attia@alexu.edu.eg
4 Faculty of Engineering and IT, University of Technology Sydney, Sydney, NSW 2007, Australia; khaled.aboufarw@uts.edu.au
5 Faculty of Computers and Artificial Intelligence, Cairo University, Cairo 12613, Egypt; a.abobakr@fci-cu.edu.eg
* Correspondence: m.hossny@unsw.edu.au
† These authors contributed equally to this work.

Abstract: Continuous action spaces impose a serious challenge for reinforcement learning agents. While several off-policy reinforcement learning algorithms provide a universal solution to continuous control problems, the real challenge lies in the fact that different actuators feature different response functions due to wear and tear (in mechanical systems) and fatigue (in biomechanical systems). In this paper, we propose enhancing the actor-critic reinforcement learning agents by parameterising the final layer in the actor network. This layer produces the actions to accommodate the behaviour discrepancy of different actuators under different load conditions during interaction with the environment. To achieve this, the actor is trained to learn the tuning parameter controlling the activation layer (e.g., Tanh and Sigmoid). The learned parameters are then used to create tailored activation functions for each actuator. We ran experiments on three OpenAI Gym environments, i.e., `Pendulum-v0`, `LunarLanderContinuous-v2`, and `BipedalWalker-v2`. Results showed an average of 23.15% and 33.80% increase in total episode reward of the `LunarLanderContinuous-v2` and `BipedalWalker-v2` environments, respectively. There was no apparent improvement in `Pendulum-v0` environment but the proposed method produces a more stable actuation signal compared to the state-of-the-art method. The proposed method allows the reinforcement learning actor to produce more robust actions that accommodate the discrepancy in the actuators' response functions. This is particularly useful for real life scenarios where actuators exhibit different response functions depending on the load and the interaction with the environment. This also simplifies the transfer learning problem by fine-tuning the parameterised activation layers instead of retraining the entire policy every time an actuator is replaced. Finally, the proposed method would allow better accommodation to biological actuators (e.g., muscles) in biomechanical systems.

Keywords: continuous control; deep reinforcement learning; actor-critic; DDPG

check for updates

Citation: Hossny, M.; Iskander, J.; Attia, M.; Saleh, K.; Abobakr, A. Refined Continuous Control of DDPG Actors via Parametrised Activation. *AI* **2021**, 2, 464–476. https://doi.org/10.3390/ai2040029

Academic Editors: Andrea Prati, Carlos A. Iglesias, Luis Javier García Villalba and Vincent A. Cicirello

Received: 11 August 2021
Accepted: 22 September 2021
Published: 29 September 2021

1. Introduction

Deep reinforcement learning (DRL) was used in different domains and achieved good results on different tasks, such as robotic control, natural language processing, and biomechanical control of digital human models [1–4].

While DRL is proven to handle discrete problems effectively and efficiently, continuous control remains a challenging task. This is because it relies on physical systems that are prone to noise due to wear and tear, overheating, and altered actuator response function depending on the load each actuators bears; this is more apparent in robotic and biomechanical control problems. In the robotic control domain, for instance, bipedal robots robustly perform articulated motor movements in complex environments and with limited resources. These robust movements are achieved using highly sophisticated model-based controllers. However, the motor characteristics are highly dependent on the load and the interaction with the environment. Thus, adaptive continuous control is required to adapt to new situations.

Biomechanical modelling and simulation present a clearer example. In a biomechanical system, the human movement is performed using muscle models [5,6]. These models simulate muscle functions, which are complex and dependent on multiple parameters, such as muscle maximum velocity, muscle optimal length, and muscle maximum isometric force, to name a few [7].

The common challenge with training DRL agents on continuous action spaces is the flow of the gradient update throughout the network. The current state of the art is relying on a single configuration of the activation function producing the actuation signals. However, different actuators exhibit different transfer functions, and also, noisy feedback from the environment propagates through the entire actor neural network; thus, a drastic change is imposed on the learned policy. The solution we are proposing in this work is to use multiple actuation transfer functions that allow the actor neural network to adaptively modify the actuation response functions to the needs of each actuator.

In this paper, we present a modular perspective of the actor in actor-critic DRL agents and propose modifying the actuation layer to learn the parameters defining the activation functions (e.g., Tanh and Sigmoid). It is important to emphasise the difference between parameterised action spaces and parameterised activation functions. In reinforcement learning, a parametrised action space is commonly referred to as a discrete action space that has one or more accompanying continuous parameters [8]. It was used to solve problems such as the RoboCup [9], which is a robots world cup soccer game [10]. On the other hand, parameterised activation functions, such as PReLU [11] and SeLU [12], were introduced to combat overfitting and saturation problems. In this paper, we propose parameterised activation functions to improve the performance of the deep deterministic policy gradient (DDPG) to accommodate the complex nature of real-life scenarios.

The rest of this paper is organised as follows. Related work is discussed in Section 2. The proposed method is presented in Section 3. Experiments and results are presented in Section 4 and discussed in Section 5. Finally, Section 6 concludes and introduces future advancements.

2. Background

Deep deterministic policy gradient (DDPG) is a widely adopted deep reinforcement learning method for continuous control problems [13]. A DDPG agent relies on three main components: the actor, critic, and experience replay buffer [13].

In the actor-critic approach [14], the actor neural network reads observations from the environment and produces actuation signals. After training, the actor neural network serves as the controller, which allows the agent to navigate the environment safely and to perform the desired tasks. The critic network assesses the anticipated reward based on the current observation and the actor's action. In control terms, the critic network serves as a black-box system identification module, which provides guidance for tuning the parameters of a PID controller. The observations, actions, estimated reward, and next-state observation are stored as an experience in a circular buffer. This buffer serves as a pool of experiences, from where samples are drawn to train the actor and the critic neural networks to produce the correct action and estimate the correct reward, respectively.

There are different DDPG variations in the literature. In [15], a twin delay DDPG (TD3) agent was proposed to limit overestimation by using the minimum value between a pair of critics instead of one critic. In [16], it was proposed to expand the DDPG as a distributed process to allow better accumulation of experiences in the experience replay buffer. Other off-policy deep reinforcement learning agents such as soft actor-critic (SAC), although relying on stochastic parameterisation, are inspired by DDPG [17,18]. In brief, SAC adapts the reparameterisation trick to learn a statistical distribution of actions from which samples are drawn based on the current state of the environment.

DDPG Challenges

Perhaps the most critical challenge of the DDPG, and off-policy agents in general, is its sample inefficiency. The main reason behind this challenge is that the actor is updated depending on the gradients calculated during the training of the critic neural network. This gradient is noisy because it relies on the outcome of the simulated episodes. Therefore, the presence of outlier scenarios impact the training of the actor, and thus, constantly change the learned policy instead of refining it. This is the main reason off-policy DRL training algorithms require maintaining a copy of the actor and critic neural networks to avoid divergence during training.

While radical changes in the learned policy may provide a good exploratory behaviour of the agent, it does come at the cost of requiring many more episodes to converge. Additionally, it is often recommended to have controllable exploratory noise parameters separated from the policy either by inducing actuation noise such as Ornstein–Uhlenbeck [19] or maximising the entropy of the learned actuation distribution [17,18]. Practically, however, for very specific tasks, and most continuous control tasks, faster convergence is often a critical aspect to consider. Another challenge, which stems from practical applications, is the fact that actuators are physical systems and are susceptible to having different characterised transfer functions in response to the supplied actuation signals. These characterisation discrepancies are almost present in every control system due to wear and tear, fatigue, overheating, and manufacturing factors. While minimal discrepancies are easily accommodated with a PID controller, they impose a serious problem with deep neural networks. This problem, in return, imposes a serious challenge during deployment and scaling operations.

3. Proposed Method

To address the aforementioned challenges, we propose parameterising the final activation function to include scaling and translation parameters k, x_0. In our case, we used $\tanh(kx - kx_0)$ instead of $\tanh(x)$ to allow the actor neural network to accommodate the discrepancies of the actuator characteristics by learning k and x_0. The added learnable parameters empower the actor with two additional degrees of freedom.

3.1. Modular Formulation

In a typical DRL agent, an actor consists of several neural network layers. While the weights of all layers collectively serve as a policy, they do serve different purposes based on their interaction with the environment. The first layer encodes observations from the environment, and thus, we propose to call it the observer layer. The encoded observations are then fed into several subsequent layers, and thus, we call them the policy layers. Finally, the output of the policy layers are usually fed to a single activation function. Throughout this paper, we denote the observer, policy, and action parts of the policy neural network as π^O, π^P, π^A, respectively. We also denote the observation, the premapped action space, and the final action space as O, $\tilde{A}$ and A, respectively. To that end, the data flow of the observation $o_t \in O$ through the policy π to produce an action $a_t \in A$ can be summarised as;

$$\begin{aligned} a_t &= \pi(o_t) = \pi^A \circ \pi^P \circ \pi^O(o_t), && (1) \\ &= \pi^A\left(\pi^P\left(\pi^O(o_t)\right)\right), && (2) \end{aligned}$$

where $\pi^O : O \to \tilde{A}$, $\pi^P : \tilde{O} \to \tilde{A}$ and $\pi^A : \tilde{A} \to A$.

In a typical actor neural network, there is no distinction between the observer and policy layers. Also, the actuation layer is simply regarded as the final activation function $\pi^A(x) = \tanh(x)$, and thus, the actor is typically modelled as one multilayer perceptron neural network (MLP). The problem with having π^A as tanh (Sigmoid is also a popular activation function where $A = [0, 1]$.) is that it assumes that all actuators in the system exhibit the same actuation-torque characterisation curves under all conditions.

3.2. Parameterising π^A

Because actuation characterisation curves differ based on their role and interaction with the environment, using a single activation function forces the feedback provided by the environment to propagate throughout the gradients of the entire policy. Therefore, we chose to use a parameterised $\pi^A(kx - kx_0)$ to model the scaling and the translation of the activation function, and thus, the data flow in Equation (2) can be expanded as;

$$k_t = \pi^K\left(\pi^P\left(\pi^O(o_t)\right)\right), \tag{3}$$

$$a_t^0 = \pi^{a_0}\left(\pi^P\left(\pi^O(o_t)\right)\right), \tag{4}$$

$$a_t = \pi^A\left(k_t\pi^P\left(\pi^O(o_t)\right) - k_t a_t^0\right), \tag{5}$$

where π^{a_0}, π^k are simple fully connected layers and π^A remains an activation function (i.e., tanh), as shown in Figure 1. Adjusting the activation curves based on the interaction with the environment allows the policy to remain intact, and thus, leads to a more stable training as discussed in the following section.

While the automatic differentiation engines are capable of adjusting the flow of gradient updates, there are two implementation considerations to factor in the design of the actor. Firstly, the scale degree of freedom parameterised by k, in the case of tanh and sigmoid, does affect the slope of the activation function. A very small $k < 0.1$ will render π^A to be almost constant while a very high $k > 25$ produces a square signal. Both extreme cases impose problems to the gradient calculations. On one hand, a constant signal produces zero gradients and prevents the policy neural network from learning. On the other hand, a square signal produces unstable exploding gradients. Another problem also occurs when $k < 0$, which usually changes the behaviour of the produced signals. Therefore, we recommend using a bounded activation function after π^k when estimating k_t.

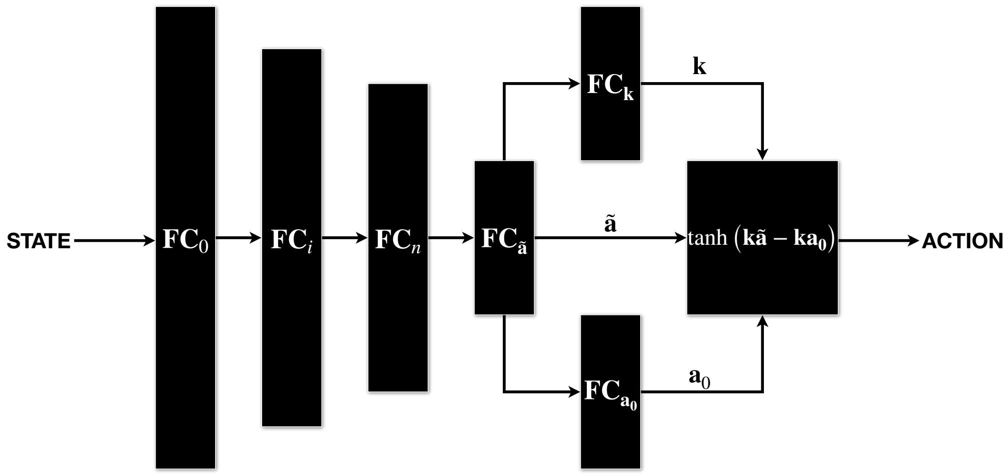


Figure 1. Proposed modification to actor. Final fully connected layer branches into two fully connected layers to learn x_0 and k parameters of $\tanh(kx - kx_0)$.

Secondly, the translation degree of freedom parameterised by a^0, allows translating the activation function to an acceptable range, which prevents saturation. However, this may, at least theoretically, allow the gradients of the policy π^P and observer π^O layers to have monotonically increasing gradients as long as the a_t^0 can accommodate. This in return may cause an exploding gradient problem. To prevent this problem, we recommend employing weight normalisation after calculating the gradients [20].

4. Experiments and Results

To test the efficacy and stability of the proposed method, we trained a DDPG agent with and without the proposed learnable activation parameterisation. Both models were trained and tested on three OpenAI gym environments, shown in Figure 2, that are `Pendulum-v0`, `LunarLanderContinous-v2`, and `BipedalWalker-v2`. For each environment, six checkpoint models were saved (best model for each seed). The saved models were then tested for 20 trials with new random seeds (10 episodes with 500 steps each). The total number of test episodes is 1,200 for each environment. The results of the three environments are tabulated in Tables 1 and 2.

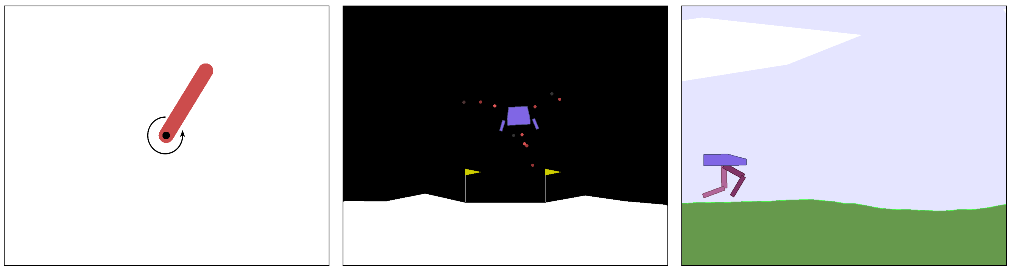

(**a**) Inverted Pendulum (**b**) Lunar Lander (**c**) Bipedal Walker

Figure 2. OpenAI gym environments used for testing.

4.1. Models and Hyperparameters

The action mapping network is where the proposed and classical models differ. The proposed model branches the final layer of into two parallel fully connected layers to infer the parameters of k, x_0 in $\tanh(kx - kx_0)$ activation function. The classical model adds two more fully connected layers separated by tanh activation function. The added layers ensures that the number of learnable parameters is the same in both models to guarantee a fair comparison.

Both models were trained on the three environments for the same number of episodes (200 steps each). However, number of steps may vary depending on early termination cases. The models were trained with 5 different pseudo-random number generator (PRNG) seeds. We set the experience replay buffer to 10^6 samples. We chose ADAM optimiser for the back-propagation optimisation and set the learning rate of both the actor and the critic to 1E-3 with first and second moments set to 0.9, 0.999, respectively. We set the reward discount $\gamma = 0.99$ and the soft update of the target neural networks $\tau = 0.005$. We also added a simple Gaussian noise with $\sigma = 0.25$ to allow exploration. During the training we saved the best model (i.e., checkpoint). DDPG hyper-parameters tuning is thoroughly articulated in [13].

Table 1. Episode Reward (mean ± std). Higher mean is better.

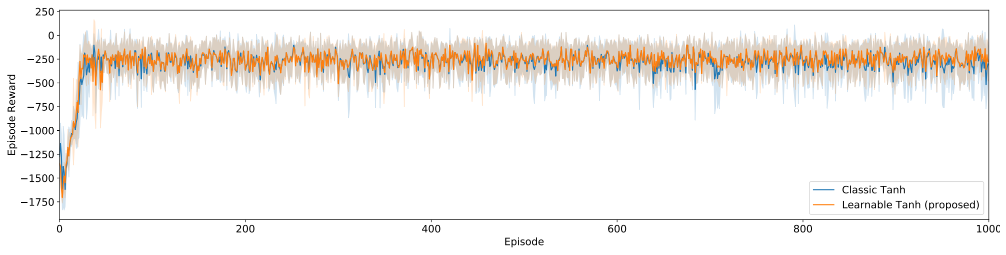
	Pendulum-v0	**LunarLanderContinuous-v2**	**BipedalWalker-v2**
Classic Tanh	−269.20 ± 167.43	114.25 ± 41.20	125.27 ± 15.78
Learnable Tanh (proposed)	−268.39 ± 166.32	**140.69** ± 45.21	**167.60** ± 8.45
Improvement	0.30%	23.15%	33.80%

Table 2. Step Reward (mean ± std). Higher mean is better.

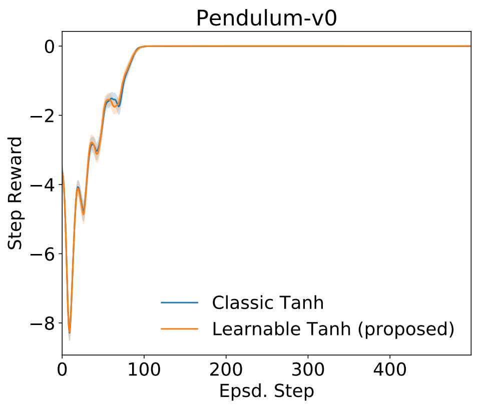
	Pendulum-v0	**LunarLanderContinuous-v2**	**BipedalWalker-v2**
Classic Tanh	−0.54 ± 0.34	0.40 ± 0.28	0.23 ± 0.09
Learnable Tanh (proposed)	−0.54 ± 0.34	**0.67** ± 0.35	**0.29** ± 0.16
Improvement	0.26%	65.59%	26.76%

4.2. Inverted Pendulum Results

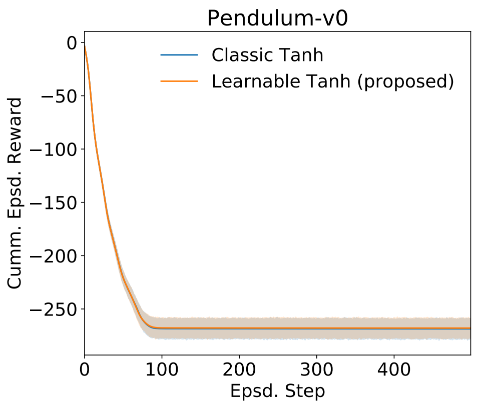
In the inverted pendulum problem (Figure 3), the improvement is marginal because the environment featured only one actuator. However, the policy adapted by the proposed agent features a fine balance of actuation signals. In contrast, the classical MLP/Tanh model exerts additional oscillating actuation signals to maintain the achieved optimal state, as shown in Figure 3e. This oscillation imposes a wear and tear challenge on mechanical systems and fatigue risks in biomechanical systems. While this difference is reflected with minimal difference in the environment reward, it is often a critical decision to make in practical applications.



(**a**) Training Performance

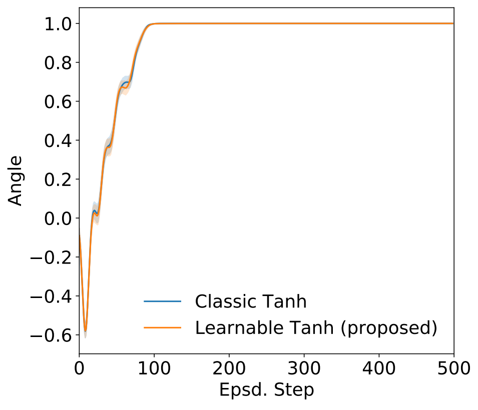


(**b**) Step Reward

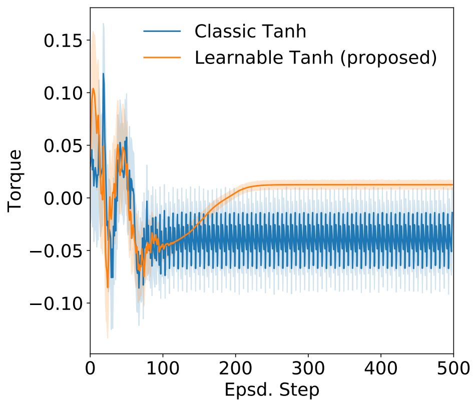


(**c**) Epsd. Reward

Figure 3. *Cont.*



(**d**) Observation

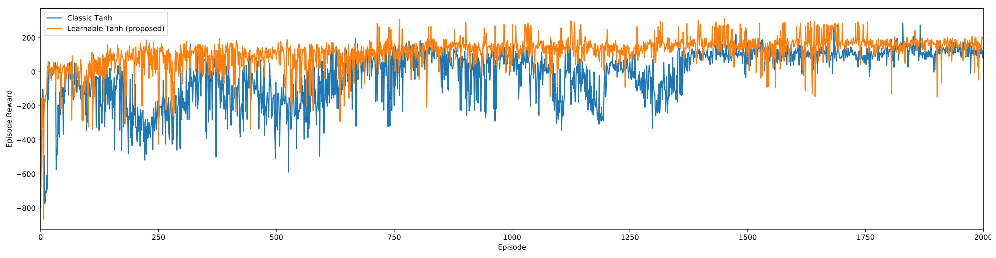


(**e**) Action

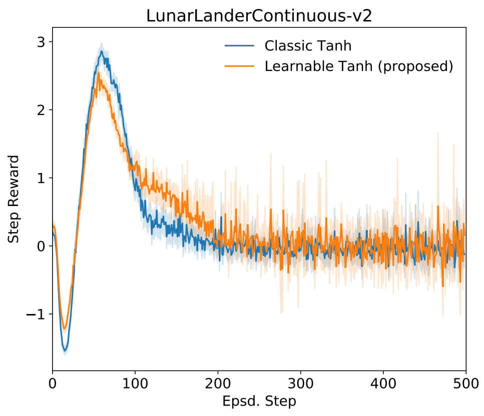
Figure 3. Inverted pendulum results. Both methods show similar training performance curves (**a**). Best models from both methods reported similar reward progression patterns (**b**–**d**). Proposed method achieves a more stable control, whereas classic method oscillates actions to maintain control (**e**).

4.3. Lunar Lander Results

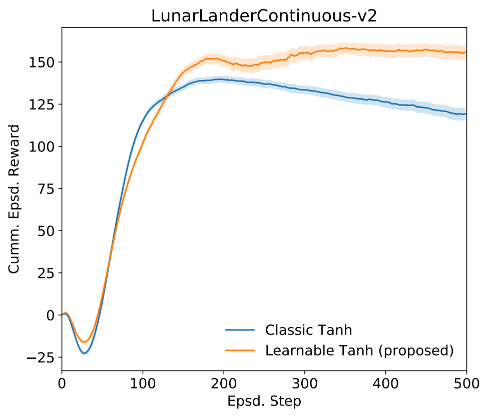
Figure 4 shows the training and reward curves of the lunar landing problem. The instant reward curve of the lunar landing problem demonstrates an interesting behaviour in the first 100 steps. The classic method adopts an energy-conservative policy by shutting down the main throttle and engaging in free falling for 25 steps to a safe margin, and then keeps hovering above ground to figure out a safe landing. The conserved energy contributes to the overall reward at each time step. While this allows for faster reward accumulation, this policy becomes less effective with different initial conditions. Depending on the speed and the attack angle, the free-falling policy requires additional effort for manoeuvring the vehicle to the landing pad. The proposed agent, on the other hand, accommodates the initial conditions and slows down the vehicle in the opposite direction to the entry angle to maintain a stable orientation, and thus, allows for a smoother lateral steering towards a safe landing as shown in Figure 5a,c. Both agents did not perform any type of reasoning or planning. The main difference is the additional degrees of freedom the proposed parametrised activation function offers. These degrees of freedom allow the proposed actor neural network to adopt different response functions to accommodate the initial conditions.



(**a**) Training Performance



(**b**) Step Reward

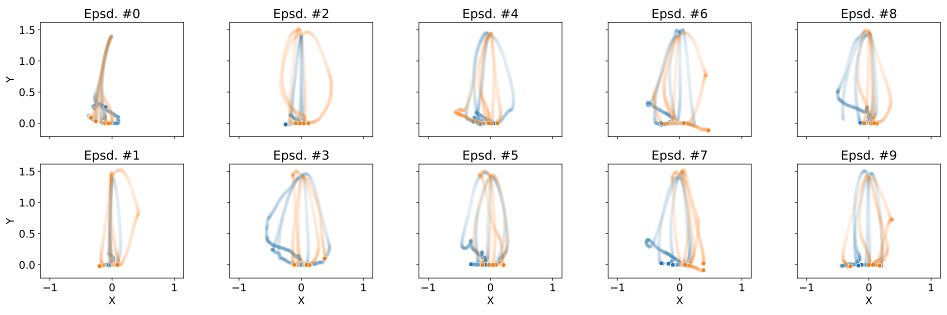


(**c**) Epsd. Reward

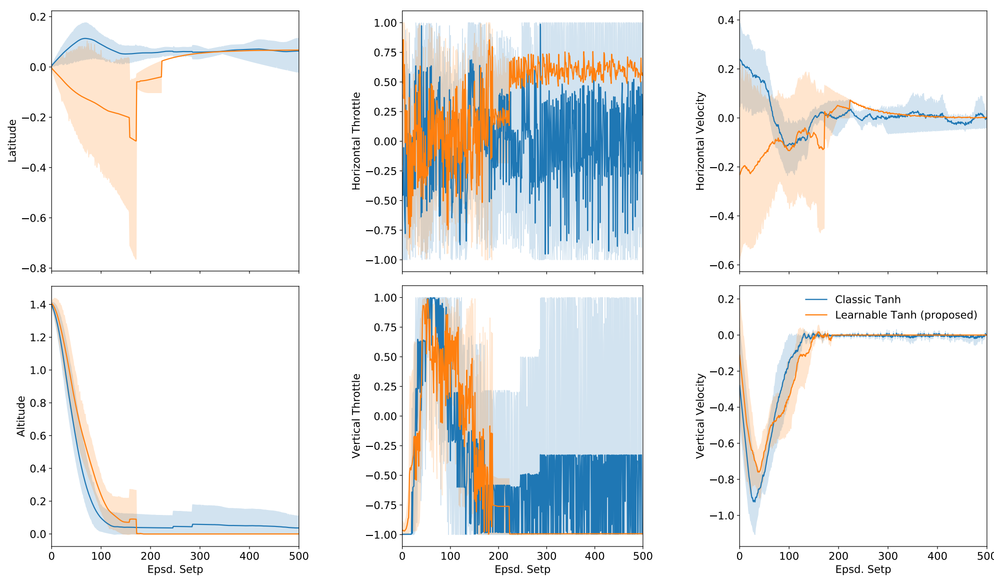
Figure 4. Lunar lander results. Proposed method converges faster to a solution and does not suffer from reward drops due to policy changes (**a**). Agents with proposed method outperform classical method in terms of reward progression (**b**,**c**).

4.4. Bipedal Walker Results

Training and reward curves of the bipedal walking problem are illustrated in Figure 6. In general, the agent with the proposed action mapping outperforms the classical agent in the training step and episode reward curves as shown in Figure 6a–c. The spikes in the step reward curves show instances where agents lost stability and failed to complete the task. The episode reward curve shows that the proposed method allows the agent to run and complete the task faster. This is due to a better coordination between the left- and right-leg while taking advantage of the gravity to minimise the effort. This is demonstrated in Figure 6d, where the proposed agent maintains a pelvis orientation angular velocity and vertical velocity close to zero. This, in return, dedicates the majority of the spent effort towards moving forward. This is also reflected in Figure 6e, where the actuation of the proposed agent stabilises faster around zero, and thus, exploits the gravity force. In contrast, the classical agent spends more effort to balance the pelvis, and thus, takes longer to stabilise actuation. Finally, the locomotion actuation patterns in Figure 6e demonstrate the difference between the adapted policies. The classical agent relies more on locomoting using `Knee2`, while the proposed agent provides more synergy between joint actuators. This difference in exploiting the gravity during locomotion is an essential key in successful bipedal locomotion as a "controlled falling" [21].

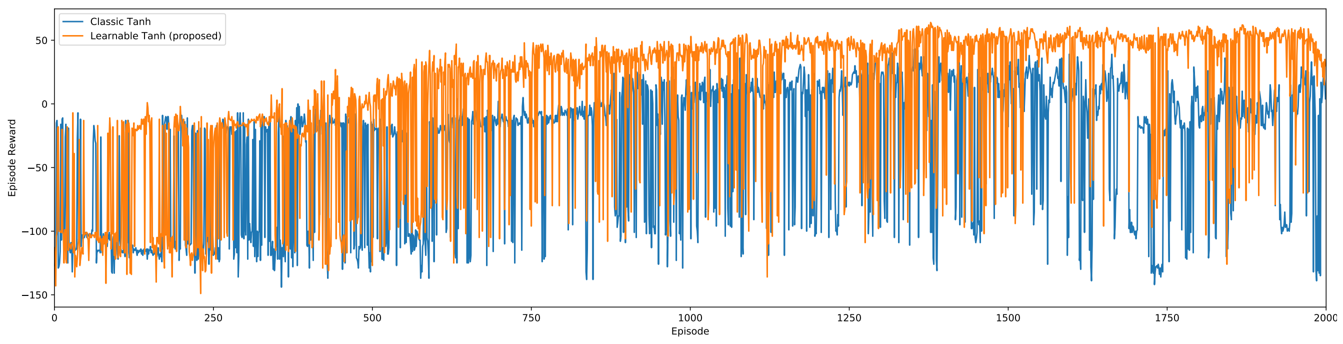


(**a**) Landing Trajectory



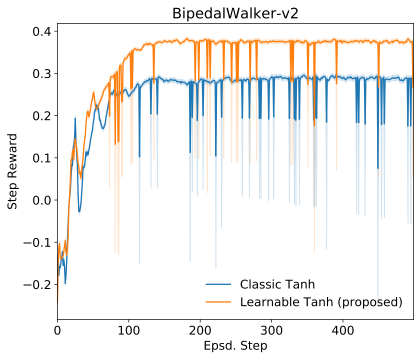
(**b**) Position (**c**) Actions (**d**) Velocity

Figure 5. Lunar lander results. Proposed and classical actors adopt different landing trajectories (**a**). Actors without the proposed method preserve effort by engaging in free falling to a safe altitude (**b**-bottom), and then exert more effort to perform safe landing (**c**-bottom). Actors with the proposed method decelerate and engage in manoeuvring to a safe landing (**b**–**d**).

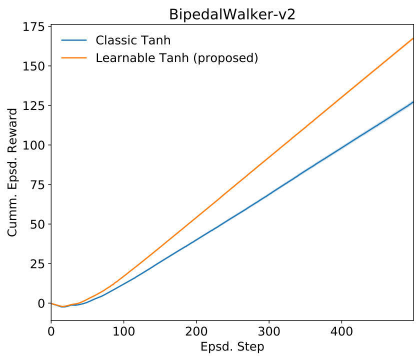


(**a**) Training Performance

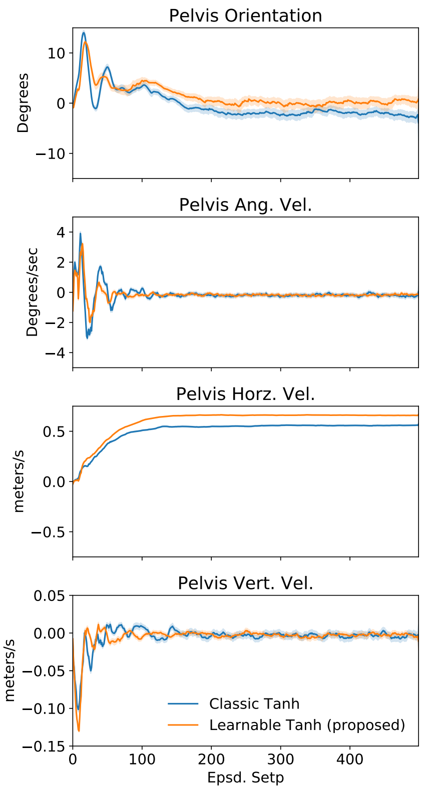
Figure 6. *Cont.*



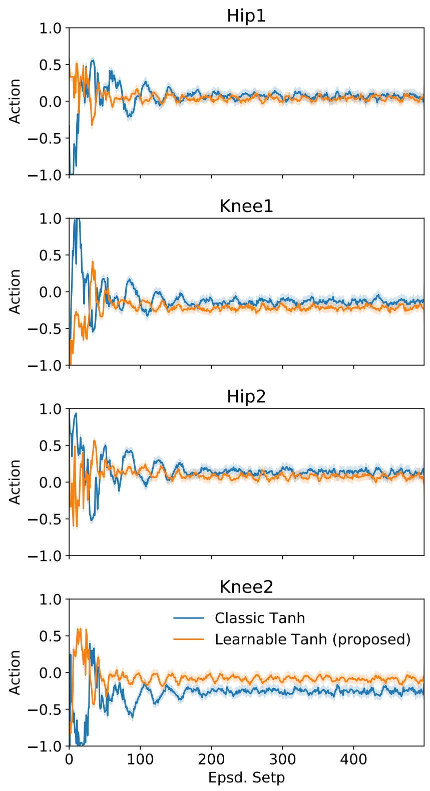
(**b**) Step Reward



(**c**) Epsd. Reward



(**d**) Observations



(**e**) Actions

Figure 6. Bipedal walker results. Proposed method records higher episode reward (**a**). Actors with proposed method outperform classical method in terms of reward progression (**b**,**c**), are more stable (**d**), and perform the task faster (**d**) and with minimal effort (**e**).

5. Discussion

The main advantage of introducing a parameterised activation function for the actuation layer in an actor neural network is minimising policy change. This is achieved by unlocking two degrees of freedom in the the activation functions (i.e., translation x_0 and scaling k), and thus, isolating control refinement from policy training. This is demonstrated by the training performance curves of the `LunarLanderContinuous-v2` and in the `BipedalWalker-v2` problems in Figures 4a and 6a, respectively. For example, in the `LunarLanderContinuous-v2` problem, the average performance of the state-of-the-art method dropped by 100% between episodes 125 and 250 while searching for the best policy. A similar pattern occurred again with less magnitude between episodes 1125 and 1250. On the other hand, the proposed actor neural network sustained a positive average episode reward from episode 750 as shown in Figure 4a. A similar behaviour is also observed with the `BipedalWalker-v2` problem in episodes 1500–2000 yet the proposed methods performed generally better in terms of episode reward as shown in Figure 6a.

The proposed method has one major limitation that is imposed by the translation degree of freedom x_0, which adjusts the location of the tanh function on the x-axis. This allows the actor neural network to tailor the tanh activation function to accommodate a wider dynamic range of actuation produced by the policy neural network. However, in doing so, it may allow for exploding gradients during the back propagation if the environment features a wide dynamic range of scenarios. This limitation can be addressed by employing weight normalisation after calculating the gradients [20]. This problem becomes more apparent when using the proposed parameterised tanh with soft-actor-critic models (SAC) [17,18]. Because SAC models rely on randomly sampling the action from a learned statistical distribution, it is susceptible to sampling outliers, which may cause exploding gradients or steer the gradients of x_0 and k in the wrong direction.

The advantage of the proposed method in the bipedal walking problem and the wide variety of activation functions demonstrated (Figure 7) a promising potential for solving several biomechanics problems where different muscles have different characteristics and response functions, as highlighted in [22]. Applications such as fall detection and prevention [23], ocular motility and the associated cognitive load, and motion sickness [24–31], as well as intent prediction of pedestrians and cyclists [32,33]. Training stability, when using the parameterised tanh in an actor-critic architecture, can potentially be used for advancing Generative Adversarial Networks (GANs) research for image synthesis [34].

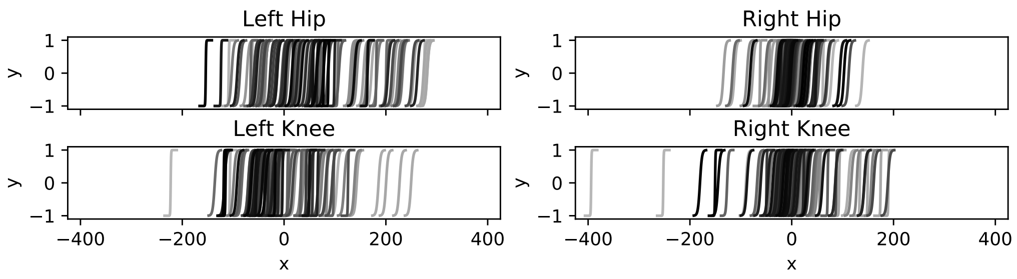


Figure 7. Resulting parameterisation of the $\tanh(kx - kx_0)$ activation function. Allowing extra degrees of freedom empowers actor neural network to accommodate outlier scenarios with minimal update to actual policy. Results here are from proposed actor trained and tested on bipedal walker environment. Colour brightness indicates different stages throughout episode from start (bright) to finish (dark).

6. Conclusions

In this paper, we discussed the advantages of adding learnable degrees of freedom to the actor in the DDPG actor-critic agent. The proposed method is simple and straightforward, yet it outperforms the classical actor, which utilises the standard tanh activation function. The main advantage of the proposed method lies in producing stable actuation signals, as demonstrated in the inverted pendulum and bipedal walker problems. Another advantage that was apparent in the lunar landing problem is the ability to accommodate changes in initial conditions. This research highlights the importance of a parameterised activation functions. While the discussed advantage may be minimal for the majority of supervised learning problems, they are essential for dynamic problems addressed by reinforcement learning. This is because reinforcement learning methods, especially the off-policy ones, rely on previous experiences during training. These advantages do allow for more stability in deploying DRL models in critical applications, such as nuclear engineering [35–38].

This research can be expanded in several directions. Firstly, the parameterisation of tanh can be extended from being deterministic (presented in this paper) to a stochastic parameterisation by inferring the distributions of k and x_0. Secondly, the separation between the policy and the action parts of the proposed actor neural network allows preserving the policy part while fine tuning only the action part to accommodate actuator characterisation discrepancies due to wear and tear during operations. Finally, the modular characterisation of different parts of the proposed actor neural network into observer, policy, and action parts requires investigating scheduled training to lock and unlock both parts alternatively to further optimise the dedicated function each part of the actor carries out.

Author Contributions: Conceptualisation, M.H., J.I.; original draft preparation, M.H., J.I.; review and editing, M.A., K.S. and A.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

DRL	Deep Reinforcement Learning
DDPG	Deep Deterministic Policy Gradient
SAC	Soft Actor-Critic
GAN	Generative Adversarial Network

References

1. Kidziński, Ł.; Ong, C.; Mohanty, S.P.; Hicks, J.; Carroll, S.; Zhou, B.; Zeng, H.; Wang, F.; Lian, R.; Tian, H.; et al. Artificial Intelligence for Prosthetics: Challenge Solutions. In *The NeurIPS'18 Competition*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 69–128.
2. Kidziński, Ł.; Mohanty, S.P.; Ong, C.F.; Hicks, J.L.; Carroll, S.F.; Levine, S.; Salathé, M.; Delp, S.L. Learning to run challenge: Synthesizing physiologically accurate motion using deep reinforcement learning. In *The NIPS'17 Competition: Building Intelligent Systems*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 101–120.
3. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529. [CrossRef] [PubMed]
4. Kober, J.; Bagnell, J.A.; Peters, J. Reinforcement learning in robotics: A survey. *Int. J. Robot. Res.* **2013**, *32*, 1238–1274. [CrossRef]
5. Thelen, D.G. Adjustment of muscle mechanics model parameters to simulate dynamic contractions in older adults. *J. Biomech. Eng.* **2003**, *125*, 70–77. [CrossRef] [PubMed]
6. Millard, M.; Uchida, T.; Seth, A.; Delp, S.L. Flexing computational muscle: Modeling and simulation of musculotendon dynamics. *J. Biomech. Eng.* **2013**, *135*, 021005. [CrossRef]
7. Zajac, F.E. Muscle and tendon: Properties , models, scaling and application to biomechanics and motor control. *Crit. Rev. Biomed. Eng.* **1989**, *17*, 359–411.
8. Masson, W.; Ranchod, P.; Konidaris, G. Reinforcement learning with parameterized actions. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, Phoenix, AR, USA, 12–17 February 2016.
9. Kitano, H.; Asada, M.; Kuniyoshi, Y.; Noda, I.; Osawa, E.; Matsubara, H. RoboCup: A challenge problem for AI. *AI Mag.* **1997**, *18*, 73.
10. Hausknecht, M.; Stone, P. Deep reinforcement learning in parameterized action space. *arXiv* **2015**, arXiv:1511.04143.
11. He, K.; Zhang, X.; Ren, S.; Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In Proceedings of the 2015 International Conference on Computer Vision, ICCV, Santiago, Chile, 7–13 December 2015; pp. 1026–1034. [CrossRef]
12. Klambauer, G.; Unterthiner, T.; Mayr, A.; Hochreiter, S. Self-Normalizing Neural Networks. *arXiv* **2017**, arXiv:1706.02515.
13. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2015**, arXiv:1509.02971.
14. Sutton, R.S.; Barto, A.G. *Introduction to Reinforcement Learning*; MIT Press: Cambridge, UK, 1998; Volume 2
15. Fujimoto, S.; Van Hoof, H.; Meger, D. Addressing function approximation error in actor-critic methods. *arXiv* **2018**, arXiv:1802.09477.
16. Barth-Maron, G.; Hoffman, M.; Budden, D.; Dabney, W.; Horgan, D.; Dhruva, T.; Muldal, A.; Heess, N.; Lillicrap, T. Distributed distributional deterministic policy gradients. *arXiv* **2018**, arXiv:1804.08617.
17. Haarnoja, T.; Zhou, A.; Abbeel, P.; Levine, S. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In Proceedings of the International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; pp. 1861–1870.
18. Haarnoja, T.; Zhou, A.; Hartikainen, K.; Tucker, G.; Ha, S.; Tan, J.; Kumar, V.; Zhu, H.; Gupta, A.; Abbeel, P.; et al. Soft actor-critic algorithms and applications. *arXiv* **2018**, arXiv:1812.05905.
19. Uhlenbeck, G.E.; Ornstein, L.S. On the Theory of the Brownian Motion. *Phys. Rev.* **1930**, *36*, 823–841. [CrossRef]
20. Salimans, T.; Kingma, D. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *arXiv* 2016, arXiv:1602.07868.
21. Novacheck, T.F. The biomechanics of running. *Gait Posture* **1998**, *7*, 77–95. [CrossRef]
22. Hossny, M.; Iskander, J. Just Don't Fall: An AI Agent's Learning Journey Towards Posture Stabilisation. *AI* **2020**, *1*, 286–298. [CrossRef]
23. Abobakr, A.; Hossny, M.; Nahavandi, S. A Skeleton-Free Fall Detection System From Depth Images Using Random Decision Forest. *IEEE Syst. J.* **2018**, *12*, 2994–3005. [CrossRef]

24. Iskander, J.; Hossny, M.; Nahavandi, S.; Del Porto, L. An Ocular Biomechanic Model for Dynamic Simulation of Different Eye Movements. *J. Biomech.* **2018**, *71*, 208–216. [CrossRef]
25. Iskander, J.; Hossny, M.; Nahavandi, S. A Review on Ocular Biomechanic Models for Assessing Visual Fatigue in Virtual Reality. *IEEE Access* **2018**, *6*, 19345–19361. [CrossRef]
26. Iskander, J.; Attia, M.; Saleh, K.; Nahavandi, D.; Abobakr, A.; Mohamed, S.; Asadi, H.; Khosravi, A.; Lim, C.P.; Hossny, M. From car sickness to autonomous car sickness: A review. *Transp. Res. Part F Traffic Psychol. Behav.* **2019**, *62*, 716–726. [CrossRef]
27. Iskander, J.; Hanoun, S.; Hettiarachchi, I.; Hossny, M.; Saleh, K.; Zhou, H.; Nahavandi, S.; Bhatti, A. Eye behaviour as a hazard perception measure. In Proceedings of the Systems Conference (SysCon), 2018 Annual IEEE International, Vancouver, BC, Canada, 23–26 April 2018; pp. 1–6.
28. Attia, M.; Hettiarachchi, I.; Hossny, M.; Nahavandi, S. A time domain classification of steady-state visual evoked potentials using deep recurrent-convolutional neural networks. In Proceedings of the 2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018), Washington, DC, USA, 4–7 April 2018; Volume 2018-April, pp. 766–769. [CrossRef]
29. Iskander, J.; Hossny, M. An ocular biomechanics environment for reinforcement learning. *arXiv* **2020**, arXiv:2008.05088.
30. Iskander, J.; Attia, M.; Saleh, K.; Abobakr, A.; Nahavandi, D.; Hossny, M.; Nahavandi, S. Exploring the Effect of Virtual Depth on Pupil Diameter. In Proceedings of the 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC), Bari, Italy, 6–9 October 2019; pp. 1849–1854.
31. Iskander, J.; Hossny, M.; Nahavandi, S. Using biomechanics to investigate the effect of VR on eye vergence system. *Appl. Ergon.* **2019**, *81*, 102883. [CrossRef] [PubMed]
32. Saleh, K.; Hossny, M.; Nahavandi, S. Intent Prediction of Pedestrians via Motion Trajectories Using Stacked Recurrent Neural Networks. *IEEE Trans. Intell. Veh.* **2018**, *3*, 414–424. [CrossRef]
33. Saleh, K.; Hossny, M.; Nahavandi, S. Spatio-temporal DenseNet for real-time intent prediction of pedestrians in urban traffic environments. *Neurocomputing* **2020**, *386*, 317–324. [CrossRef]
34. Attia, M.; Hossny, M.; Zhou, H.; Nahavandi, S.; Asadi, H.; Yazdabadi, A. Realistic hair simulator for skin lesion images: A novel benchemarking tool. *Artif. Intell. Med.* **2020**, *108*, 101933. [CrossRef] [PubMed]
35. Hossny, K.; Hossny, A.; Magdi, S.; Soliman, A.Y.; Hossny, M. Detecting shielded explosives by coupling prompt gamma neutron activation analysis and deep neural networks. *Sci. Rep.* **2020**, *10*, 13467. [CrossRef]
36. Hegazy, A.H.; Skoy, V.R.; Hossny, K. Optimization of Shielding—Collimator Parameters for ING-27 Neutron Generator Using MCNP5. *EPJ Web Conf.* **2018**, *177*, 02003. [CrossRef]
37. Hossny, K.; Magdi, S.; Nasr, F.; Yasser, Y.; Magdy, A. Neutron depth profile calculations using artificial neural networks. *EPJ Web Conf.* **2021**, *247*, 06046. [CrossRef]
38. Hossny, K.; Magdi, S.; Soliman, A.Y.; Hossny, A.H. Detecting explosives by PGNAA using KNN Regressors and decision tree classifier: A proof of concept. *Prog. Nucl. Energy* **2020**, *124*, 103332, doi: 10.1016/j.pnucene.2020.103332. [CrossRef]