Model-based actor-critic: GAN (model generator) + DRL (actor-critic) => AGI

Aras Dargazany (arasdar@uri.edu)

ABSTRACT

Our effort is toward unifying GAN and DRL algorithms into a unifying AI model (AGI or general-purpose AI or artificial general intelligence) which has general-purpose applications to: **(A) offline learning** (of stored data) like GAN in (un/semi-/fully-)SL setting such as big data analytics (mining) & visualization; **(B) online learning** (of real or simulated devices) like DRL in RL setting (with/out environment reward) such as (real or simulated) robotics & control; Our core proposal is adding an (generative/predictive) environment model to the actor-critic (model-free) architecture which results in a model-based actor-critic architecture with temporal-differencing (TD) error and an episodic memory. The proposed AI model is similar to (model-free) DDPG and therefore it can also be called model-based DDPG. To evaluate it, we compare it with (model-free) DDPG by applying them both to a variety (wide range) of independent simulated robotic and control task environments in OpenAI Gym and Unity ML Agents. Our initial limited experiments show that DRL and GAN in model-based actor-critic results in an incremental goal-driven intellignce required to solve each task with similar performance to (model-free) DDPG. Our future focus is to investigate the proposed AI model potential to: **(A) unify DRL field** inside AI by producing competitive performance compared to the best of model-based (PlaNet) and model-free (D4PG) approaches; **(B) bridge the gap between AI and robotics** communities by solving the important problem of reward engineering with learning the reward function by demonstration;

Introduction: GAN (generative model) vs DRL (actor-critic) – offline learning vs online learning

Currently all problems in artificial intelligence (AI) are formulated as an end-to-end (deep) artificial neural network optimization problem, also known as deep neural network learning or in short deep learning (DL)[1]. These problems are mainly classified into: offline learning and online learning; Offline learning refers to learning from the stored data or dataset which are nowadays known as big data problem. In offline learning, the stored big or small dataset might not have any labels (unsupervised), might be partially labeled (semi-supervised) or might be fully labeled (fully-supervised). Online learning refers to learning from devices or machines such as robots (simulated or real) directly in real-time to achieve a specific goal or accomplish a specific task. Online learning might be refered to as reinforcement learning (RL) or reward-based learning or robot learning or robust control learning as well. RL is sometimes refered to as deep RL (DRL) which is due to the application of DL to RL problems. Originally DL[1] was proposed as a powerful solution to offline learning, more specifically supervised learning (SL) of big data such as Imagenet classification[2] and TIMIT speech recognition[3]. Currently, generative adversarial nets (GAN)[4] & deep reinforcement learning (DRL)[5], specifically actor-critic (AC) algorithms are the fore-frontiers of DL/AI community: (A) GAN for offline learning or (un/, semi-/,fully-)supervised learning (SL)[6,7]; and (B) AC for online learning or (deep) RL or DRL[8,9]. DL formulates an optimization problem with a user-defined objective (loss or cost) function and lacks a single unifying objective (loss or cost) function[1,10]. GAN & AC both consist of two models (networks): one of which approximates the main function for prediction (actor for AC or generator for GAN) and another one approximates the objective (loss or cost) function (discriminator for GAN or critic for AC). This has revolutionized the assumptions behind DL algorithms in terms of having a pre-defined fixed objective (loss or cost) function although applying gradient descent (and ascent) to these networks (GAN & AC) often leads to mode collapse, unstable training and no convergence in some cases. AC methods[11,12] and GAN[4] are two main classes of multilevel (in fact bilevel) optimization problems with close connection[10]. Since both of these hybrid or multilevel optimization models suffer from stability issues, mode collapse, and convergence difficulties, techniques[10] for stabilizing training have been developed largely independently by the two communities. This work is mainly inspired and built upon some important previous works directly or indirectly[13–29]. In this work, we mainly focus on the two most stable one of each: Wasserstein GAN (WGAN)[30] and deep deterministic policy gradients (DDPG)[8]. The main contribution of this work is to point out: 1. the strong connection/similarity between WGAN and DDPG as the two most stable classes of GAN and AC (DRL) model; 2. Given the first contribution, we argue that DDPG demonstrates adversarial learning behavior very similar to WGAN; 3. Given the first contribution, how this similarity can lead to a model-based DDPG (model-based actor-critic); Eventually, we want to conclude that adversarial intelligence (as the product of adversarial learning process/behavior) might be a general-purpose AI (AGI) since it is applicable to both offline learning through GAN for (un-, semi-, fully-)SL and online learning through AC (DRL) for RL and robot learning.

GAN for offline learning

Generative adversarial networks (GAN)[4] formulates the unsupervised learning problem as a game between two opponents: a generator G which generates sample images from a random noise sampled from a fixed probability distribution/density function (PDF) or a fixed noise source such as normal function; and a discriminator D which classifies the sample images as real/true or fake/false (maps an image to a binary classification probability); The original or vanilla GAN[4] was formulated as a zero sum game with the cross-entropy loss between the discriminator prediction and the true identity of the image as real or generated/fake. To make sure that the generator has gradients from which to learn even when the discriminator classification accuracy is high, the generator loss function is usually formulated as maximizing the probability of classifying a generated sample as true rather than minimizing its probability of being classified false (or fake). GAN is formulated in a bilevel optimization setting.

GAN for offline learning or (un, semi-, fully-) supervised learning (learning from data)

GAN was initially applied to unsupervised learning (no label available for the dataset) for image generation, data distribution modelling and discovering the data structure[4]. Later on, it was applied to (semi-)supervised learning (SL) problem[6,7] where the data is either partially labeled (semi-supervised) or fully labeled (fully-supervised)[31] such as image classification, object detection, and object recognition. The first application of GAN to (semi-)SL problem[6] reported 64% accuracy on SVHN[1] dataset with 1% labeled data. In another original work[7] by Tim Salimans at OpenAI, GAN reaches over 94% accuracy on the same SVHN dataset using only 1,000 labeled examples out of almost 100,000 examples which means only 1% of the entire dataset is labeled. Though originally GAN was proposed as a form of generative model for unsupervised learning[4], GAN has proven useful for semi-supervised learning[7], and fully supervised learning[31]. Therefore, GAN is a potentially capable approach for offline-learning of big data such as big data analytics/mining and visualization.

GAN has no proof of convergence.

Unfortunately, there is no proof of convergence for GAN. No one has yet shown that GAN[4] will converge to an equilibrium. On contrary, very many practical experiences have shown that GAN is unstable and might lead to mode collapse which means technically no convergence (equilibrium) at all. One ray of hope is provided in the idea of using Earth Mover's metric (or Wasserstein metric)[30]. MIX-GAN[32] architecture with multiple generators and discriminators can also reach an approximate equilibrium under fairly tight conditions which are not usually practical. Generative multi-adversarial network[33] shows that multiple discriminators does indeed improve empirical performance.

Wasserstein GAN: stable GAN with Wasserstein loss

WGAN[7,30] uses Wasserstein loss (also known as earth mover) as an alternative loss function instead of the traditional/original GAN loss function, cross-entropy loss[4]. Wasserstein loss improves the stability of GAN training/learning considerably by almost eliminating the important problem of mode collapse. WGAN[7] is applied to semi-supervised learning for classification without labels (partially labelled). Using Wasserstein distance loss and some other tricks (gradient penalty), WGAN[7] achieved state-of-the-art results in semi-supervised classification on MNIST, CIFAR10 and SVHN at the time of publication. Later on, we show that Wasserstein loss provides deep connection and deep similarity with Bellman loss (TD loss using Bellman equation or Q-learning or Q-loss learning) which is used in DDPG[8] as the most stable AC method in DRL.

Self-attention GAN: GAN using attention layers

Self-Attention GAN (SAGAN) uses attention[34] for long-range dependency modeling (proven by[35]) for video prediction[36] and image generation tasks. In contrast with vanilla GAN which can generate high-resolution images with only spatially local points, SAGAN can generate higher resolution images using spatially non-local points (or global and local points at the same time as mentioned in non-local neural net[36]). Visualization of the attention layers in the generator shows the non-local regions which correspond to the object shapes are detected compared to the local regions of a fixed shape in the vanilla GAN.

DRL (actor-critic) for online learning (real-time learning by interacting with environment)

Actor-critic (AC) methods are a long-established class of techniques in reinforcement learning (RL) [2]. While most RL algorithms either focus on learning a value function like value iteration and TD-learning (value-based methods), or learning a policy directly such as policy gradient methods (policy-based methods), AC methods learn both the actor (policy) and the critic (value) simultaneously. In some AC methods, the actor model (policy function) is updated with respect to the approximate critic model (value function), in which case learning behavior and architecture similar to those in GAN can result. Formally, consider the typical Markov decision processes (MDP) setting for RL, where we have a set of states S, actions A, and discount factor [0, 1]. The aim of AC methods is to simultaneously learn an action-value function that predicts the total expected discounted

---

[1]street-view house numbers (http://ufldl.stanford.edu/the-street-view-house-numbers/)

future reward and learn a policy that is optimal for that value function. Actor-critic is also formulated in a bilevel optimization setting similar to GAN: There are many AC methods that attempt to solve this problem. The distinction between these methods lies mainly in the way training proceeds. Traditional AC methods optimize the policy through policy gradients and scale the policy gradient by Bellman loss, also known as temporal-difference (TD) loss or TD error, while the action-value function is updated by ordinary TD error learning[37]. In this work, we mainly focus on deep deterministic policy gradients (DDPG)[8] which is intended for the case where actions and observations are continuous, and use deep learning (DL) for function approximation of both the action-value function (critic) and the state-action function or policy (actor). DDPG is an established DRL (AC) approach with continuous actions which updates the policy (actor) by backpropagating the gradients of the estimated value from critic with respect to the actions rather than backpropagating the TD error directly.

Actor-critic for online learning and (deep) RL or DRL

Deep Q-learning (DQL) or deep Q-network[5] was one of the breakthrough work in RL and the beginning of deep RL (DRL). Actor-critic (AC) algorithms[38] are one of the most powerful RL or DRL algorithms which are composed of two networks: actor and critic. AC methods are models from deep reinforcement learning (DRL) in which a model learns an action-value function that predicts the expected total future reward (the Critic), and a policy that is optimal for maximizing that value (the Actor or controller). DDPG[8], as a stable DL-based AC method, is in fact the extension of DQL approach to the continuous action space since DQL was in fact applied to the Atari game for discrete action space control and search.

Actor-critic methods have proof of convergence to an optimal policy.

The original AC method[11] can be viewed as the formal beginning of work in computational reinforcement learning. The critic in AC is like the discriminator in GAN, and the actor in AC methods is like the generator in GAN. In both systems, there is a game being played between the actor (generator) and the critic (discriminator). The actor begins exploring the state space, and the critic should learn to evaluate the random exploratory behavior of the actor. It was formally proved by Konda[39–41] that in any Markov decision process (MDP), AC methods will eventually converge to the optimal policy. The proof was non-trivial and shown a decade after AC proposal[11]. Recently (35 years after AC methods proposal) Google DeepMind proposed the same AC algorithm combined with DL (DDPG[8]) to solve the difficult Atari games from raw video input.

Deep Deterministic Policy Gradients: stable AC (DRL) method with Bellman/TD loss

Deep Deterministic Policy Gradients (DDPG)[8] (stable AC approach in terms of convergence) is the application of DL to AC methods. DDPG uses Bellman loss[5] or temporal-difference (TD) loss[12] as the loss function. TD loss learning is also know as Q-learning[12] which after the introduction of DL[1] gave birth to Deep Q-learning (DQL) or deep Q-network[5] which was the beginning of deep RL (DRL). DDPG[8], as a stable DL-based AC method, is in fact the extension of DQL approach to the continuous action space since DQL was in fact applied to the Atari games for discrete action space control and search.

Connection between GAN and AC methods

Both GAN and AC methods can be seen as bilevel or two-time-scale optimization problems, meaning one model is optimized with respect to the optimum of another model. Bilevel optimization problems have been extensively studied under AC methods by Konda[39–41] mainly under the assumption that both optimization problems are linear or convex[42]. On contrary, both DDPG[8] and WGAN[30] which are the center of attention in this work and the most stable kind of GAN and AC methods, are in fact non-linear and have non-convex optimization surfaces due to DL which are non-linear function approximator. The question is though: (A) if GAN and AC are by nature the same algorithms or not? Or (B) can one emerge from another? if yes, which emerges from which? Pfau and Vinyals[10] show that GAN can be viewed as AC methods in an environment where the actor cannot affect the reward. Therefore, they answer the above questions as following: (A) Yes, they are connected; (B) Yes, GAN can emerge from AC methods; They review a number of extensions to GAN and AC algorithms with even more complicated information flow. They encourage both GAN and AC (DRL) communities to develop general, scalable, and stable algorithms for multilevel optimization with DL, and to draw inspiration across communities. Pfau and Vinyals[10] confirmed that AC and GAN are siblings and therefore there must be a parent algorithm/architecture that include both methods. Pfau and Vinyals[10] encourage us for more investigation of deeper connection (and possibly convergence and unification) between GAN and AC methods for the development and adoption of a more general-purpose AI technique (AGI) applicable as GAN or AC (DRL). Pfau and Vinyal[10] describes the connection between GAN and AC as an MDP in which GAN is a modified version of AC method as following: Consider an MDP where the actions set every pixel in an image. The environment randomly chooses either to show the image generated by the actor or show the real image. Let the reward from the environment be 1 if the environment chose the real image and 0 if not. This MDP is stateless as the image generated by the actor does not affect future data. The AC architecture learning in this environment resembles the GAN game. A few adjustments have to be made to make it identical. If the actor had access to the state, it could trivially pass a real image forward. Therefore the actor must be a

blind actor, with no knowledge of the state. Stateless MDP doesn't prevent the actor from learning though. The mean-squared Bellman/TD loss is usually used as the loss function for AC (specifically DDPG), mean-cross entropy adversarial loss is used instead for GAN (so-called GAN loss). The actor's parameters in AC should not be updated if the environment shows a real image. Critic zeros down its gradients for the actor (no update for actor's parameters/weights) if the reward is 1 for the real image. If the reward is 0 for the fake image, critic still zeros down its gradients for the actor. If the reward is 1 for the fake image, it's time for updating actor's parameters using critic's output and gradients. This is how, GAN can be seen as a modified AC method with blind actor in stateless MDP.

Is actor-critic adversarial?

According to[10], it is not obvious why an AC algorithm should lead to an adversarial behavior; Typically the actor and critic are trying to optimize complimentary loss functions (compatible or orthogonal), rather than optimize the same loss function in different directions (adversarial). The adversarial behavior in AC emerges due to the stateless MDP in which the GAN game is being played since the actor cannot have any causal effect on the reward in this stateless MDP. A critic, however, cannot learn the causal structure of the game from input examples alone, and moves in the direction of features that predict reward more accurately (minimizing the reward prediction error/loss). The actor moves in a direction to increase reward (maximizing the reward value) based on the best estimate from the critic, but this change cannot lead to an increase in the true reward, so the critic will quickly learn to assign lower value in the direction the actor has moved. Thus the updates to the actor and critic, which ideally would be orthogonal (as in compatible actor-critic or complimentary loss functions) instead becomes adversarial. Despite these differences between GAN and typical RL problems (settings), we believe there are enough similarities to generalize between GAN and AC (DRL) algorithms. This generalization might be the path toward general-purpose AI or artificial general intelligence (AGI).

Target networks is the only main difference between AC method (specifically DDPG) and GAN (specifically WGAN).

Since the action-value function appears twice in the Bellman equation, stability can be an issue in Q-learning with function approximation. Target networks address this by fixing one of the networks in TD updates or possibly slowing down the updates of this network so-called target network. This will turn the Q-learning problem from RL to SL and helps the Q-learning to converge whereas without it will most likely diverge. Since the GAN game can be seen as a stateless MDP, the second appearance of the action value function disappears. Therefore, we do not consider target networks applicable to the GAN setting[10].

Compatibility of actor-critic

According to[10], one of the unique theoretical developments of AC methods is the notion that the actor and critic are compatible or complementary in terms of loss function. It is not clear if the notion of compatibility can be naturally extended to the GAN setting. We would generally prefer our GAN to be adversarial than compatible. We hope that by pointing out the deep connections between GAN and AC (DRL), we encourage both communities of GAN and DRL (AC) to merge and join forces.

Model-based actor-critic: GAN + DRL (actor-critic)

We propose to teach machines to accomplish more complex tasks in one common environment with modelling the real world/environment with rich textures and complex structural compositions. We address three challenges in particular for training an agent to model the real-world: **First**, how to apply the adversarial learning[4] to improve the quality of the generated environment model (*eñv*) since GAN is proved to be effective in image generation and modelling data distribution tasks[43]. This is also known as explicit modelling of the environment. Actor-critic (AC) methods, specifically deep deterministic policy gradients (DDPG)[8] models the environment internally in the critic network. We chose DDPG as our default AC approach due to the continuous action space of the agent. **Second**, how to build an efficient differentiable neural renderer that can simulate the environment and is transferable to other tasks as well so that we don't need to start training the environment from scratch. Also the more tasks our agent accomplishes, the more accurate our environment model (the generator or the neural differentiable simulator/renderer) becomes. We train a generative neural network which directly maps the current action (A) and the current state of environment/world S into the next state of environment (S'). This differential renderer (generator) can be combined with AC and turn into model-based AC that can be trained in an end-to-end fashion, which might significantly boosts both the modelling quality and convergence speed in terms of solving the task. *NOTE: generator, neural simulator, differential renderer, environment modeller, and world renderer are all the same.* **Third**, how to design a reward function is another really important challenge since reinforcement learning (RL or deep RL also refered as DRL) or reward-based learning is impossible without the reward. Manually designing this reward function for the complex problems/tasks is almost impossible. Specifically, if we want to apply DRL or deep RL to the robotics problem, we have to be able to design an accurate reward function or go for

a simple task which the reward function design is fairly easy. The latter is not always possible and therefore we have to be able to also learn the reward function. This has turned into a headache for many researchers in artificial intelligence (AI) and robotics. In summary, our contributions/solutions for the three aforementioned challenges/problems are also three-fold: **1.** We approach the modelling task with the GAN and build a model-based DRL agent by combining GAN with (model-free) AC method (DDPG) that can model the agent's environment. To this end, we build a differentiable neural renderer (generator) for efficiently modelling the environment. This neural environment renderer (generator) and a discriminator model the world by training model-based DRL agent in an end-to-end fashion. Discriminator is required to discriminate between the real environment samples and fake ones (generated/rendered ones) by neural environment model. **2.** Explicitly modelling the environment with generator neural net compared to implicitly modelling it in critic of AC method (DDPG[8]) helps transferring this model to other problems and tasks in that same environment. This transfer learning makes our DRL agent very sample efficient, meaning it can accomplish the same task with much less episodes compared to (model-free) AC or original DDPG. **3.** Given the explicit transferable environment model (generator) and the target image (an image of the goal or what we want to accomplish/gain at the end), we can learn how to design the reward function with the discriminator as it becomes more and more powerful in discriminating between the real environment and explicit environment model. This is also solves the problem of designing/hand-engineering the reward function for complex tasks.

Literature

SPIRAL[44] is an adversarially trained DRL agent that learn the structures in images and tries to paint them from scratch, but fails to recover the details of images. SPIRAL++[45] is an improved version of SPIRAL which is using GAN and RL agent at the same time in a reinforced adversarial learning fashion for paining an image in a canvas from scratch. Both SPIRAL and SPIRAL++ are composed of an actor-critic and discriminator but they use a fixed/non-differentiable renderer which is generating the painting. StrokeNet[46] combines differentiable neural image/paining renderer (a generator) on top of actor-critic and discriminator using recurrent neural network (RNN) to train agents to paint but fails to generalize on color images. The most inspiring and similar work to us is done by Huang et al.[47] which is proposing a model-based DDPG which is in fact a model-based actor-critic as well. He is proposing four neural networks in his model-based DDPG agent to learn how to paint on canvas from scratch and without any external reward. He eventually experimentally compared his model-based DDPG to the original (model-free) DDPG and experimentally demonstrates the superiority of model-based DDPG.

DRL approaches
Superficially, SPIRAL and SPIRAL++ can be seen as a model-free RL techniques for stroke based, non-photorealistic rendering/generating. Similarly to some modern stroke-based rendering techniques, the positions of strokes are determined by a learned system (namely, an RL agent). Unlike traditional methods however, the objective function that determines the goodness of the output image is learned unsupervised via the adversarial objective (adversarial loss). This adversarial objective allows us to train without access to ground-truth strokes, enabling the method to be applied to domains where labels are prohibitively expensive to collect, and for it to discover surprising and unexpected styles. In this line of work, there are a number of works that use constructions similar to SPIRAL and SPIRAL++ to tune the parameters of non-differentiable/fixed simulators. In another line of work, Frans and Cheng (2018)[48], Nakano (2019)[49], Zheng et al. (2019)[46], and Huang et al. (2019)[47] achieve remarkable learning speed by relying on backpropagation/ neural model of the renderer, however they rely on being able to train an accurate model of the environment in the first instance.

Model-free vs model-based RL approaches
Recently, several works (Frans and Cheng in 2018[48], Nakano in 2019[49], Zheng et al. in 2019[46], and Huang et al. in 2019[47]) proposed to improve sample efficiency and convergence stability of generative RL agents by replacing the actual non-differentiable/fixed rendering simulator with a differentiable neural one (generator neural net) which is trained offline to predict how an action affects the state of the canvas environment. Although promising, there are three scenarios in which model-free approaches like SPIRAL and SPIRAL++ might be a better approach: **Firstly**, one of the main advantages of the neural environments is training agents by directly backpropagating the gradient coming from the objective of interest (either reconstruction loss if there is no discriminator or adversarial loss if there is a discriminator or GAN setting). Unfortunately, this means that the action space of the agent has to be continuous to efficiently use backpropagation gradient (neural net learning). This means if the action space is discrete, we might have problem in learning our neural networks in model-based approaches. **Secondly**, the success of the RL agent training largely depends on the quality of the neural model of the environment. The simulator used in SPIRAL and SPIRAL++ is arguably easy to learn since new strokes interact with the existing drawing in a fairly straightforward manner. Environments which are highly stochastic or require handling of object occlusions and lighting might pose a challenge for neural network based environment models. **Thirdly**, while in principle possible, designing a model for a simulator with complex dynamics may be a non-trivial task. The majority of the recent works relying on neural renderers

assume that the simulator state is fully represented by the appearance of the canvas and therefore only consider non-recurrent state transition models. There is no need for such an assumption in the SPIRAL and SPIRAL++ framework.

Conclusion of the literature
The use of neural simulators in model-based DRL are largely orthogonal to SPIRAL and SPIRAL++ and are likely to improve their performance further. Nonetheless model-based DRL approaches are dependent on a given target image which is not the case for SPIRAL and SPIRAL++.

Model-based actor-critic

We propose the model-based DRL framework shown in Figure 1. This framework is mainly composed of two stream
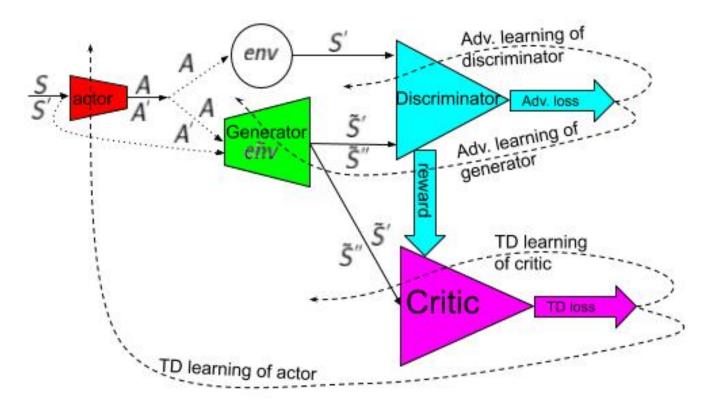
Figure 1. The overall architecture of model-based actor-critic (AC): (a) At the inference/testing stage, the actor outputs an action (A) to the real environment (*env*) and generator (*eñv*). In result, the real environment (*env*) outputs the next state (S') to the actor and to the discriminator. The discriminator applies adversarial learning on Discriminator and Generator based on the resulting adversarial loss and update the environment model (*eñv*) more accurately. (b) At the training stage, the actor and critic are being trained/updated together using critic's temporal-difference (TD) loss through reinforcement learning (Q-learning). The required reward for reinforcement learning (Q-learning) is given by the discriminator at each step, and the training samples are randomly sampled from the replay buffer. Adversarial learning (GAN learning) is also being done the same as inference/testing stage. S: current states of the environment; S': next states of the environment; A: actions; A': next actions, *env*: real environment/world of the agent; *eñv*: environment model which is being modelled by generator; $\tilde{S}'$: predicted/generated states after current states of environment; $\tilde{S}''$: predicted/generated states after next states of environment; Adv. learning: adversarial learning; TD loss: temporal-difference loss;

frameworks: GAN framework for adversarial learning stream and AC framework for reinforcement learning stream.

GAN framework for adversarial learning stream
Since GAN has been widely used because of its great ability in modelling the data distribution by measuring the distribution distance between the generated (fake) data and the target (real) data, therefore the adversarial learning stream is mainly composed of GAN, a generator and discriminator, with an actor as you can see in figure 2. Wasserstein GAN (WGAN)[30] is our
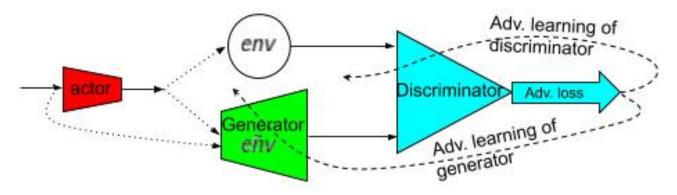
Figure 2. Adversarial learning stream in model-based actor-critic architecture

preference compared the original GAN[4] since it is an improved version of the original GAN[4] which uses the *Wasserstein – l* distance, also known as Earth-Mover distance. The objective of the discriminator in WGAN is defined as 1:

$$\max_{D} \mathbb{E}_{y \sim \mu}[D(y)] - \mathbb{E}_{x \sim \nu}[D(x)] \tag{1}$$

where D denotes the discriminator, ν and μ are the fake samples and real samples distribution. The adversarial learning stream is responsible for two important tasks: **1.** modeling the environment using the generator as accurately as possible. Discriminator in this stream helps discriminating between the real environment and the environment model *eñv*. Using a generator neural network to model the environment has two advantages: **First**, it is transferable to any other tasks in that same environment and also the more tasks we do with this environment model the more accurate it will become in terms of predicting the next state of the environment $st + 1$ given the current state st and current taken action at; **Second**, it boosts the performance of the agent in terms of successful completion of the task and sample-efficiency which in simple words means completing the task in much less episodes of try-and-error; The training samples can also be generated randomly using Computer Graphics rendering programs. The generator (environment model *eñv*) can be quickly trained with supervised learning on the GPU/s. The model-based transition dynamics $st + 1 = eñv(st, at)$ and the reward function $r(st, at)$ are differentiable neural network. The generator is a deep (end-to-end) neural network consisting of several fully connected layers and convolution layers.

2. learning the reward function for the reinforcement learning stream since the reinforcement learning stream is impossible without the the reward function. Using a discriminator neural network to model the reward has also two advantages: **First**, it is adaptable to other tasks in that same environment and we don't have to manually engineer/design the reward function; **Second**, it boosts the performance of the agent in terms of successful completion of the task by accurately learning the appropriate reward function for that specific task; The discriminator similar to generator is also a deep (end-to-end) neural network consisting of several fully connected layers and convolution layers but the output is a scalar value/score $L(st)$. The conditional GAN training schema[50] is suggested by[47] as the reward metric, where fake samples and the target vs real samples and the target are used for reward learning as shown in Figure 3 taken from[47]. we want to reduce the differences between the current state and target state as much as possible. To achieve this, we set the difference of discriminator scores from st to $st + 1$ using equation 2 as the reward for guiding the learning of the actor.

$$r(st, at) = L(st) - L(st + 1) \tag{2}$$

0.1 Actor-critic framework for DRL

The reinforcement learning stream is mainly composed of the actor-critic model and the generator which contain the model of the environment *eñv* as shown in figure 4. The optimization of the RL agent using the model-based AC is different from that using the original AC. At timestep t, the critic takes $st + 1$ as input instead of both of st and at as shown in figure 5. The critic still predicts the expected reward $V(st)$ for the state st but without the need for current action as the input also shown in figure 5. The new expected reward $V(st + 1)$ is a value function $V(st)$ trained using discounted reward: $V(st) = r + \gamma V(st + 1)$ Here $r(st, at)$ is the reward when performing action at based on st. The environment function $st + 1 = eñv(st, at)$ is the generator network which models the environment.

Model parameters and variables

We model a task as an MDP with a state space S an action space A, a reward function $r(st, at)$, an environment function $env(st, at)$ which can be the real environment *env* or the environment model *eñv*. The details of these components are as
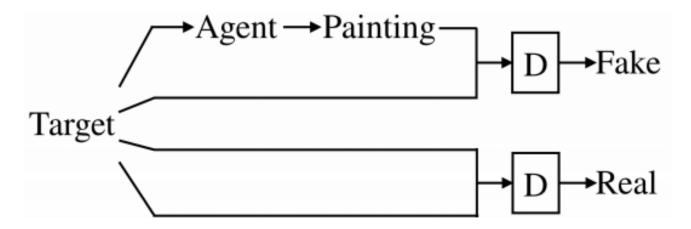
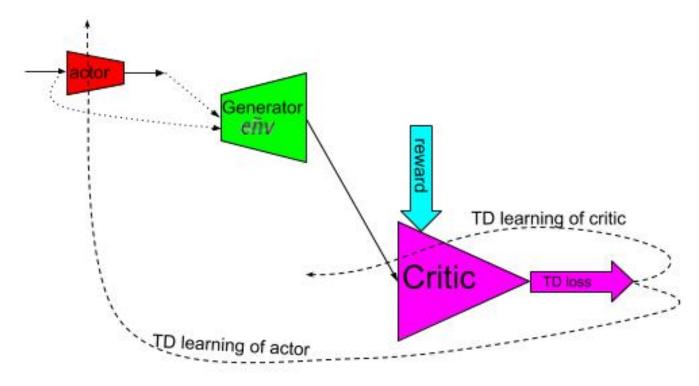**Figure 3.** This is the training paradigm for discriminator according to Huang[47].

Figure 4. Reinforcement learning stream in model-based actor-critic architecture

following: **The state space** includes all information regarding the state of the environment which can be observed by the agent. **The environment function** $st + 1 = env(st, at)$ makes the transition process from the current state of the environment to the next state. **The action space** at of the agent is a set of continuous parameters that control the position and orientation of the agent or part of the agent required for completing the target task. We define the behavior of an agent as a policy function π that maps states to deterministic actions ($\pi : S \rightarrow A$). **Timesteps** t is every step that the agent make an observation of the environment state st and takes an action accordingly at and in result of the taken action, the current state of the environment evolves/transitioned based on the environment function $st + 1 = env(st, at)$ which can be the real one env or the fake one $e\tilde{n}v$. **Reward** Selecting a suitable metric to measure the difference between the current state and the target state is crucial for RL agent. In fact, reinforcement learning is reward-based learning and without a stable reward function/metric it is not possible. The reward is defined as follows: $r(st, at) = L(st) - \gamma L(st + 1)$ where $r(st, at)$ is the reward at step t, $L(st)$ is the measured loss between the target environment state I and the current state st and $L(st + 1)$ is the measured loss between the target environment state I and the next state $st + 1$. $L(st)$ and $L(st + 1)$ is formulated as the discriminator output score for current environment

state st and the next state $st + 1$. To reach the target environment state I, the agent should model the environment accurately for an precise prediction of the next state (adversarial learning stream using GAN) and maximize the cumulative rewards (reinforcement learning stream using AC) in one episode.

Why model-based actor-critic?

Since the action space in the control and robotic tasks are mainly continuous and high dimensional, discretizing the action space to adapt to discrete DRL methods such as deep Q-network (DQN)[5] and policy gradients (PG)[12] is quite burdensome if possible. In contrast, deterministic policy gradients (DPG)[51] uses deterministic policy to resolve the difficulties caused by high-dimensional continuous action space. If the environment states are also high-dimensional, (original) DDPG[8] solves this problem too using deep (end-to-end) convolutional neural network (DNN) so-called deep learning (DL)[1,2]. In the original DDPG, there are two networks: the actor $\pi(s)$ and critic $Q(s,a)$. The actor models a policy π that maps a state st to action at. The critic estimates the expected reward for the agent taking action at at state st, which is trained using Bellman equation 3 as in Q-learning[52] and the data is sampled from an experience replay buffer:

$$Q(st,at) = r(st,at) + \gamma Q(st+1, \pi(st+1)) \tag{3}$$

Here $r(st,at)$ is a reward given by the environment when performing an action at at the state st. The actor $\pi(st)$ is trained to maximize the critic's estimated value $Q(st, \pi(st))$. In other words, the actor decides an action/policy for each state. Based on the current state and the target state, the critic predicts an expected reward for the action. The critic is optimized to estimate more accurate expected rewards. We cannot train a good-performance RL agent using original DDPG because it's hard for the agent to model the complex environment well composed of real-world images/observations during learning. The World Model[53] is a method to make agent understand the environments effectively. Similar to[47–49], we design a neural model of the environment (refered to as neural renderer in the cited works) so that the agent can model the environment effectively. Then it can explore the modelled environment and improve its policy efficiently. The difference between the two algorithms visually shown in Figure 5 from[47] is very similar to what we are proposing.
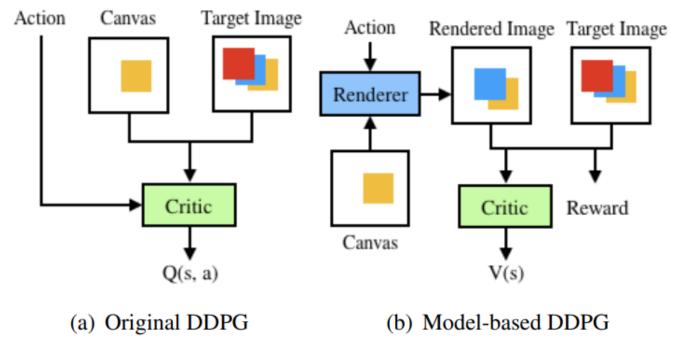

Figure 5. According to Huang[47]: In the original actor-critic (DDPG[8]), critic learns an implicit model of the environment. In the model-based actor-critic, the environment is explicitly modeled by a neural renderer (generator in our work *eñv*), which helps to train an agent efficiently.

Experiments: Initial limited experimental results

We implemented and tested the proposed model-based actor-critic in some simulated environments such as OpenAI Gym and Unity ML agents that simulates a number of independent tasks in their own environment which provides both sensors input and

the reward function as visualized in figure 6. These two simulators provide a number of tasks in their own unique environments that varies from classical control problems (e.g. CartPole), robotic problems (e.g. Reacher arm), and famous video games (e.g. Car race). These task environments are independent from each other in a sense that the knowledge from one is not required nor can be transferred to another one. The inputs and output of the AI model (agent) and the task environment (env) are visually shown in figure 6 for better understanding of the data flow in between the AI model (agent) and the environment task (env) and how the experiments are being performed.
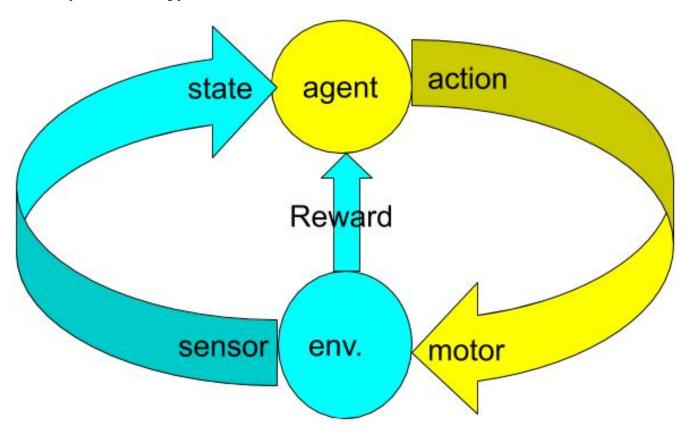

Figure 6. The inputs and outputs of the AI model and the task environment; The reward is how we can bridge the gap between agent & environment; The reward is also how we can define the task/goal for the robot; agent: the proposed AI model which is based on model-based actor-critic, env: environment which is the task is being presented to the agent;

Reacher environment simulated by Unity ML-agents as a robotic arm problem

Reacher is one of the Unity ML-agents environments for deep reinforcement learning (DRL) research experiments. Reacher environment features and specifications are listed as following: Reacher is a double-jointed arm which can move to target locations; **Goal**: The agents must move its hand to the goal location, and keep it there; **Agent Reward Function (independent)**: For each step, agent's hand reaches the goal location, it receives +0.1; **State space:** A vector of 26 variables corresponding to position, rotation, velocity, and angular velocities of the two arm rigid bodies; **Action space:** A vector of 4 continuous variables corresponding to torque applicable to two joints; **To solve this task, Benchmark Mean Reward: 30** Reacher environment is capable of using one agents or 20 agents (multi-agents) (figure 7). We have implemented the proposed model-based actor-critic for initially experimentation on solving such tasks (figure 6) using PyTorch library which is a python-based deep learning framework for Facebook. The initial results of applying the model-based actor-critic to the Reacher environment with one agent or multi-agents (twenty) are shown in figure 8. Based on the solving criteria of Reacher, the proposed model-based actor-critic solved this task with one agent in roughly 500 episodes and with twenty agents (multi-agent) in roughly 175 episodes as shown in figure 8. We conducted this experiment to make sure that we can implement the proposed architecture or it is implementable. We also wanted to make sure that it works in terms of solving reinforcement learning tasks.
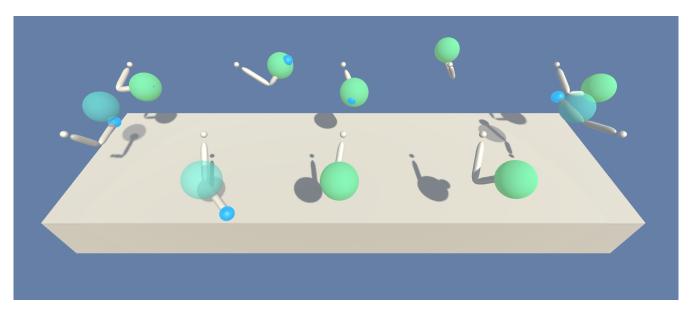
Figure 7. The Reacher environment in Unity, capable of using one or multi-agents experimental platform.
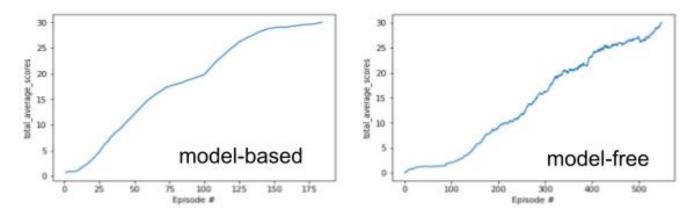

Figure 8. The current performance of the proposed approach in one Reacher and multiple Reachers environment; Y-axis is the total average score (average accumulated rewards) over the number of episodes on X-axis; We believe model-based actor-critic can reduce the number of episodes (improve sample efficiency) compared to original (model-free) actor-critic.

Conclusion & future perspective

Our limited experiments show that deep reinforcement learning (DRL) and GAN in (our AGI model) can result in an incremental goal-driven intellignce required to potentially solve (general-purpose) variety of independent tasks, each in their own separate independent environments. Our future focus is to investigate:

- the connection between the model-based actor-critic (DDPG) and the brain: is model-based DDPG architecture and learning compatible and plausible with the brain?

- the connection between GAN and the brain: is there any adversarial learning in the brain?

- the application of GAN to (semi-/fully-) SL problems for offline learning of the stored data for big data analytics (mining) & visualization: is it applicable to all variety of SL problems?

- the application of model-based actor-critic (DDPG) to variety of independent tasks in only one same environment with reward (or reward function) such as DeepMind control suite: can it transfer skill from one task to another?

- the application of model-based actor-critic (DDPG) for (simulated or real) robotic control without reward signal from the environment: can we learn reward function instead of manually engineering/designing it in a robotic environment?

Connection between model-based actor-critic and the brain

The proposed Model-based Actor-Critic algorithm/architecture is the most sensible/reasonable way of combining of GAN and DRL (AC algorithms). I am reasoning that GAN can be used to model the environment/surroundings/our world as a generative model and in a realistic form. And DRL (AC algorithms) can be used to to perform/come up with the set of actions which maximize the future reward in order to perform a task composed of multiple skills or accomplish a goal. Neuroscientifically speaking, I highly believe this is how our brain functions or this is a systematic architecture of our brain (shown in figure 9 A): Cerebellum (CBL) can be the actor, Cortex (CTX) can be the generator to generate/model the environment and Basal ganglia (BG) can be the discriminator (indirect path) & critic (direct path). The work on this topic is very important in terms of building AGI models, combining model-free and model-based RL algorithms by figuring out how to combine GAN with DRL in a sensible/reasonable way. A similar computational model of the brain is also illustrated in figure 10.
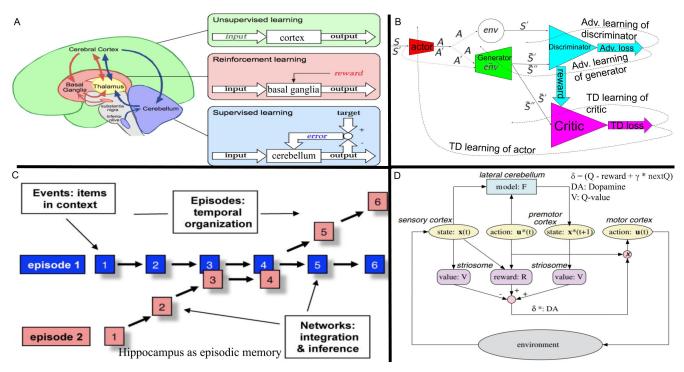

Figure 9. The connection between the model-based DDPG (actor-critic) and brain: (A) the recent findings of the brain architecture (integrated network of BG, CBL, CTX, and hippocampus[54,55]) and brain learning paradigm (super-learning[56]); (B) The proposed model-based actor-critic (DDPG) which might help us toward reaching a unifying AI model (AGI) which is more plausible and compatible with the brain; (C) Hippocampus role and functionality in connection with this architecture biologically and artificially according to[57]; (D) An old computational paradigm similar to our proposal according to Doya[54,58] which is the computational architecture of model-based action selection combines forward models in the CBL and reward predictors in the BG proposed by Doya in 1999[58];

Hippocampus role in DRL systems

Figure 9 C is a conceptual illustration of hippocampus as a short-term episodic memory space containing: (A) Temporal events within a specific context in which they occurred. These episodes are composed of temporal organization of events such as links between events and episodes; (B) Spatial navigation or events as a spatial memory according to Eichenbaum (2004)[57]; According to this recent article https://singularityhub.com/how-to-build-a-mind-this-learning-theory-may-hold-the-answer/, Kumaran et al.[60] updates the complementary learning systems (CLS) theory proposed by and McClelland et al.[61], which holds that intelligent agents must possess two learning systems, instantiated in mammalians in neocortex and hippocampus. The CLS theory states that the brain relies on two memory systems that allow it to rapidly soak in new information, while maintaining a structured model of the world that's resilient to noise. The core principles of the CLS theory is the understanding of memory in biological systems according to Kumaran et al.. In 1995, McClelland et al. observed a ground-breaking memory phenomenon in patients with damage to their hippocampus which was the fact that they could no longer form new memories although their past memories and concept were untouched and retrievable. This led a land-mark paper by McClelland et al. in which they proposed CLS theory which in fact introduces hippocampus as a short-term memory (replay buffer or episodic memory) and cortex as
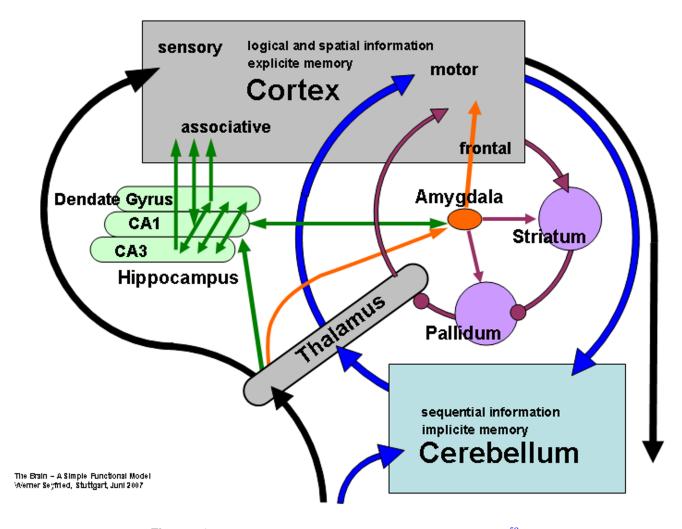
Figure 10. A similar old brain architecture proposed in this model[59]

long-term memory. According to Bendor[62], the hippocampus does not fully replay all the recent activation patterns. Instead, it picks the most rewarding events and selectively replays them to the cortex. This 'replay' has been postulated to be important for memory consolidation. Task-related cues can enhance memory consolidation when presented during a post-training sleep session, and, if memories are consolidated by hippocampal replay, a specific enhancement for this replay should be observed. Bendor[62] also indicates that this replay during sleep can be manipulated by external stimulation which provide further evidence for the role of hippocampal replay in memory consolidation. This might also means that rare but meaningful events might be prioritized for cortical learning, this is also known prioritized replay in DRL literature as well applied to D4PG[9] compared to DDPG[8].

Potential to unify DRL field inside AGI community

Our proposed AGI model has the potential to unify DRL field inside AI community by producing competitive performance compared to the best of model-based (PlaNet[63] and model-free (D4PG[9]) approaches in DRL since our proposed AGI model architecture adds an environment model network on top of model-free architecture (DDPG as an AC model). This environment model network functions as a long-term memory of environment since it models the environment and can predict the next state of the environment given its current state and the action applied to it. Our proposed AGI model is based on policy network (actor), value network (critic), and model network (env. model or generator or generated env.) which makes it model-based, policy-based, and value-based all at the same time. We should be able to compare our proposed AGI model with the best of model-based (PlaNet[63] and model-free (D4PG[9]) in the new DeepMind simulated environment (DeepMind Control Suite[64]) which is built to benchmark model-based (and model-free) DRL algorithms. This environment[64] includes variety of tasks in one environment and under one environmental condition which makes it much more suitable to evaluate and compare DRL algorithms compared to OpenAI Gym and Unity ML-agents environments which are composed of variety of independent task

environments with their own different independent environment conditions.

Towards learning the reward function by demonstration to bridge the gap between AI & robotics

The goal is learning-by-demonstration for the reward function problem or reward engineering problem. The main problem in applying AI to robotics is the reward (and punishment) signal. Hand-engineering the reward function for a goal-oriented/driven behavior is a very difficult and sometimes impossible problem. Learning-by-demonstration[65] helps us approach this problem using proposed AGI architecture. The GAN network help us learn a task by the demonstration of a task[65]. Using Q-learning (and the Q-network or critic network or value network) As shown in figure 1, we can generate the reward by the discriminator network in every time-step for the robot similar to figure 3 proposed by Huang[47]. We have two different kinds of task for the robots: (a) the task we know how to do it so that we can demonstrate for the robot; (b) The task we don't know how to do it but we know how it looks like when it is done or accomplished; Either one of them needs a reward function so that it can quantify the progress and the direction for the robot in terms of solving the task. Therefore designing such reward function is the most crucial gap between robotics and AI. Recently there have been some successful developments in the state of this art (reviewed by Amarjyoti[66]). Our proposed AGI model tackles this problem or at least approaches this problem by learning the Q-network (critic network) which is related to defining reward function for the robot. We can learn this reward function by demonstration of a user if the user knows how to demonstrate the task. In case we don't know how to do the task, we should be able to provide some preliminary data to the AGI model so that it knows how the final state of environment looks like in terms success or failure. The latter one sounds much more complicated than what we think it is and it needs much more investigation.

References

1. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *nature* **521**, 436 (2015).

2. Krizhevsky, A., Sutskever, I. & Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105 (2012).

3. Hinton, G. *et al.* Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal processing magazine* **29** (2012).

4. Goodfellow, I. *et al.* Generative adversarial nets. In *Advances in neural information processing systems*, 2672–2680 (2014).

5. Mnih, V. *et al.* Human-level control through deep reinforcement learning. *Nature* **518**, 529 (2015).

6. Kingma, D. P., Mohamed, S., Rezende, D. J. & Welling, M. Semi-supervised learning with deep generative models. In *Advances in neural information processing systems*, 3581–3589 (2014).

7. Salimans, T. *et al.* Improved techniques for training gans. In *Advances in neural information processing systems*, 2234–2242 (2016).

8. Lillicrap, T. P. *et al.* Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).

9. Barth-Maron, G. *et al.* Distributed distributional deterministic policy gradients. *arXiv preprint arXiv:1804.08617* (2018).

10. Pfau, D. & Vinyals, O. Connecting generative adversarial networks and actor-critic methods. *arXiv preprint arXiv:1610.01945* (2016).

11. Barto, A. G., Sutton, R. S. & Anderson, C. W. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, cybernetics* 834–846 (1983).

12. Sutton, R. S., McAllester, D. A., Singh, S. P. & Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, 1057–1063 (2000).

13. Lotfi, M., Solimani, A., Dargazany, A., Afzal, H. & Bandarabadi, M. Combining wavelet transforms and neural networks for image classification. In *2009 41st Southeastern Symposium on System Theory*, 44–48 (IEEE, 2009).

14. Dolatabadi, M. D., Dargazany, A. & Nouri, H. 2d-visualization of 3d medical images within a distributed system: A short survey. In *Proceedings of the World Congress on Engineering and Computer Science*, vol. 1 (2014).

15. Dolatabadi, M. D., Dargazany, A. & Nouri, H. Optimized web based method for 2d-visualization of 3d medical images. In *Proceedings of the World Congress on Engineering and Computer Science*, vol. 1, 6 (2014).

16. Gyllinsky, J. V. *et al.* Anear: Android electronically activated recorder-a smartwatch-based data collection & analysis framework for connected health. *2017 Rhode Island NIH IDeA Symp.* (2017).

17. Dargazany, A. & Soleimani, A. Hand tracking using kernel density approximation. In *2009 International Conference on Artificial Intelligence and Computational Intelligence*, vol. 3, 142–146 (IEEE, 2009).

18. Dargazany, A. & Solimani, A. Kernel-based hand tracking. *Aust. J. Basic Appl. Sci.* **3**, 4017–4025 (2009).

19. Dargazany, A., Soleimani, A. & Ahmadyfard, A. Multibandwidth kernel-based object tracking. *Adv. Artif. Intell. (16877470)* (2010).

20. Dargazany, A. & Nicolescu, M. Human body parts tracking using torso tracking: applications to activity recognition. In *2012 Ninth International Conference on Information Technology-New Generations*, 646–651 (IEEE, 2012).

21. Dargazany, A. & Berns, K. Stereo-based terrain traversability estimation using surface normals. In *ISR/Robotik 2014; 41st International Symposium on Robotics*, 1–7 (VDE, 2014).

22. Dargazany, A. & Berns, K. Terrain traversability analysis using organized point cloud, superpixel surface normals-based segmentation and pca-based classification. In *Workshop on Field and Assistive Robotics* (2014).

23. Dargazany, A. R. Iterative temporal differencing with random synaptic feedback weights support error backpropagation for deep learning. *arXiv preprint arXiv:1907.07255* (2019).

24. Dargazany, A. R., Stegagno, P. & Mankodiya, K. Wearabledl: Wearable internet-of-things and deep learning for big data analytics—concept, literature, and future. *Mob. Inf. Syst.* **2018** (2018).

25. Dargazany, A. R. Deep learning research landscape & roadmap in a nutshell: past, present and future–towards deep cortical learning. *arXiv preprint arXiv:1908.02130* (2019).

26. Dargazany, A. R., Abtahi, M. & Mankodiya, K. An end-to-end (deep) neural network applied to raw eeg, fnirs and body motion data for data fusion and bci classification task without any pre-/post-processing. *arXiv preprint arXiv:1907.09523* (2019).

27. Dargazany, A. R. Human body parts tracking: Applications to activity recognition. *arXiv preprint arXiv:1907.05281* (2019).

28. Dargazany, A. R. Mlr (memory, learning and recognition): A general cognitive model–applied to intelligent robots and systems control. *arXiv preprint arXiv:1907.05553* (2019).

29. Dargazany, A. R. Stereo-based terrain traversability analysis using normal-based segmentation and superpixel surface analysis. *arXiv preprint arXiv:1907.06823* (2019).

30. Arjovsky, M., Chintala, S. & Bottou, L. Wasserstein generative adversarial networks. In *International conference on machine learning*, 214–223 (2017).

31. Isola, P., Zhu, J.-Y., Zhou, T. & Efros, A. A. Image-to-image translation with conditional adversarial networks. *arxiv* (2016).

32. Arora, S., Ge, R., Liang, Y., Ma, T. & Zhang, Y. Generalization and equilibrium in generative adversarial nets (gans). In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 224–232 (JMLR. org, 2017).

33. Durugkar, I., Gemp, I. & Mahadevan, S. Generative multi-adversarial networks. *arXiv preprint arXiv:1611.01673* (2016).

34. Vaswani, A. *et al.* Attention is all you need. In *Advances in neural information processing systems*, 5998–6008 (2017).

35. Ramsauer, H. *et al.* Hopfield networks is all you need. *arXiv preprint arXiv:2008.02217* (2020).

36. Wang, X., Girshick, R., Gupta, A. & He, K. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 7794–7803 (2018).

37. Sutton, R. S. Learning to predict by the methods of temporal differences. *Mach. learning* **3**, 9–44 (1988).

38. Crites, R. H. & Barto, A. G. An actor/critic algorithm that is equivalent to q-learning. In *Advances in Neural Information Processing Systems*, 401–408 (1995).

39. Konda, V. R. & Tsitsiklis, J. N. Actor-critic algorithms. In *Advances in neural information processing systems*, 1008–1014 (2000).

40. Konda, V. Actor-critic algorithms (ph. d. thesis). *Dep. Electr. Eng. Comput. Sci. Mass. Inst. Technol.* (2002).

41. Konda, V. R. & Tsitsiklis, J. N. On actor-critic algorithms. *SIAM journal on Control. Optim.* **42**, 1143–1166 (2003).

42. Colson, B., Marcotte, P. & Savard, G. An overview of bilevel optimization. *Annals operations research* **153**, 235–256 (2007).

43. Ledig, C. *et al.* Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4681–4690 (2017).

44. Ganin, Y., Kulkarni, T., Babuschkin, I., Eslami, S. & Vinyals, O. Synthesizing programs for images using reinforced adversarial learning. *arXiv preprint arXiv:1804.01118* (2018).

45. Mellor, J. F. *et al.* Unsupervised doodling and painting with improved spiral. *arXiv preprint arXiv:1910.01007* (2019).

46. Zheng, N., Jiang, Y. & Huang, D. Strokenet: A neural painting environment. In *Proceedings of the IEEE conference on learning and representation (ICLR)* (2018).

47. Huang, Z., Heng, W. & Zhou, S. Learning to paint with model-based deep reinforcement learning. *arXiv preprint arXiv:1903.04411* (2019).

48. Frans, K. & Cheng, C.-Y. Unsupervised image to sequence translation with canvas-drawer networks. *arXiv preprint arXiv:1809.08340* (2018).

49. Nakano, R. Neural painters: A learned differentiable constraint for generating brushstroke paintings. *arXiv preprint arXiv:1904.08410* (2019).

50. Isola, P., Zhu, J.-Y., Zhou, T. & Efros, A. A. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1125–1134 (2017).

51. Silver, D. *et al.* Deterministic policy gradient algorithms. In *Proceedings of MLR* (2014).

52. Watkins, C. J. & Dayan, P. Q-learning. *Mach. learning* **8**, 279–292 (1992).

53. Ha, D. & Schmidhuber, J. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems*, 2450–2462 (2018).

54. Caligiore, D. *et al.* Consensus paper: towards a systems-level view of cerebellar function: the interplay between cerebellum, basal ganglia, and cortex. *The Cerebellum* **16**, 203–229 (2017).

55. Bostan, A. C. & Strick, P. L. The basal ganglia and the cerebellum: nodes in an integrated network. *Nat. Rev. Neurosci.* **19**, 338 (2018).

56. Caligiore, D., Arbib, M. A., Miall, R. C. & Baldassarre, G. The super-learning hypothesis: Integrating learning processes across cortex, cerebellum and basal ganglia. *Neurosci. & Biobehav. Rev.* (2019).

57. Eichenbaum, H. & Cohen, N. J. Can we reconcile the declarative memory and spatial navigation views on hippocampal function? *Neuron* **83**, 764–770 (2014).

58. Doya, K. What are the computations of the cerebellum, the basal ganglia and the cerebral cortex? *Neural networks* **12**, 961–974 (1999).

59. Seyfried, W. The brain - a simple functional model. http://www.werner-seyfried.de/brain/.

60. Kumaran, D., Hassabis, D. & McClelland, J. L. What learning systems do intelligent agents need? complementary learning systems theory updated. *Trends cognitive sciences* **20**, 512–534 (2016).

61. McClelland, J. L., McNaughton, B. L. & O'Reilly, R. C. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychol. review* **102**, 419 (1995).

62. Bendor, D. & Wilson, M. A. Biasing the content of hippocampal replay during sleep. *Nat. neuroscience* **15**, 1439 (2012).

63. Hafner, D. *et al.* Learning latent dynamics for planning from pixels. *arXiv preprint arXiv:1811.04551* (2018).

64. Tassa, Y. *et al.* Deepmind control suite. *arXiv preprint arXiv:1801.00690* (2018).

65. Singh, A., Yang, L., Hartikainen, K., Finn, C. & Levine, S. End-to-end robotic reinforcement learning without reward engineering. *arXiv preprint arXiv:1904.07854* (2019).

66. Amarjyoti, S. Deep reinforcement learning for robotic manipulation-the state of the art. *arXiv preprint arXiv:1701.08878* (2017).