Humanoid Can Walk:
A Data-Driven Reinforcement Learning Method

Lihan Zha

Weiyang College, Tsinghua University

Abstract: Training humanoid robot to walk steadily has always been a open challenge. This work mainly refers to DeepMimic and proposes a data-driven reinforcement learning method and adopts PPO as the algorithm. Through an innovative design of reward functions that smartly encourages the robot to mimic the target poses and execute the command at the same time, the humanoid robot is able to walk and turn its direction in a very steady and flexible manner.

Keywords: Humanoid robot; Reinforcement learning; Data-driven method; Proximal policy optimazation (PPO)

1 Introduction

Humanoid robots are designed for many purposes, especially experimental purposes, such as the research on bipedal locomotion [1]. Since human body has an extremely sophisticated structure, modelling the motion of humans remains a challenging problem, and for the time being, very few methods exist that can simulate the diversity of behaviors exhibited in the real world. Persistent challenges in this field include generalization [2]. Methods that rely on manually de-signed controllers produce convincing results , but their ability to generalize to new skills and situations is limited by the availability of human insight.

It turns out that data-driven methods perform best in dealing with current obstacles, especially those combined with reinforcement learning (RL). Value iteration methods are used to fulfil a certain task based on given motion clips [3]. GAIL in 2016 used motion data to design rewards of RL tasks [4]. DeepMimic in 2018 defines both an imitation and a task-specific reward together to enable a humanoid robot to imitate reference motions as well as fulfilling specific tasks [5].

This work mainly refers to DeepMimic, training a humanoid robot to walk steadily in a simulation environment. The core idea is to design a reward function that both represents the similarity between the reference clip and the robot's motion, and how well the robot answers the command. The basic RL algorithm adopted in this work is proximal policy optimization (PPO) and vanilla policy gradient (VPG), whose performances are compared. The prototype of the humanoid robot is from ISRLab, the lab where I'm working in now.

2 Methodology

Overall, this work is formulated as a classical reinforcement learning problem. The model receives a set of reference motion clips (in the form of array) , that specifies the robot's target poses $\{p_t(t)\}$ at each frame of a single walking cycle, and a velocity command $\{c(t)\}$, that specifies the direction and speed of the robot, as inputs $\{I(t)\}$. At each frame, the similarity between robot's real pose $p(t)$ and target pose $p_t(t)$ defines the reward $r_I(t)$ (I represents 'imitation'), the similarity between $p(t)$ and $c(t)$ defines the reward $r_C(t)$ (C represents 'command'), and the total reward $r(t) = \omega^I r_I(t) + \omega^C r_C(t)$. The inputs, together with the states of the robot, are mapped to an action $a(t)$ by a neural network policy $\pi(a(t)|s(t), I(t))$. The action $a(t)$ specifies the target angles of the

joints, which are used to compute the torques applied by PD controllers. The ultimate goal is to find an optimal policy π^* in order to maximize the total reward $\sum_t r(t)$.

2.1 Reinforcement Learning Algorithms

In this work, 2 reinforcement learning algorithms are adopted and their performances and compared, respectively VPG and PPO, and the main model is designed based on PPO. VPG is solely used for ablation study because it is the very naïve version of policy gradient algorithms, the category that PPO in essence falls in. This work does not include deep Q-network (DQN) for ablation study because DQN is too hard to tune in this setting; more importantly, DQN (and its variants) converges slower than PG -style methods, which is very fatal because simulation is very consuming in this work, often taking 1 day or more to finish one training task.

2.1.1 Vanilla Policy Gradient (VPG)

VPG is a classic reinforcement learning algorithm. Denote robot's action as a_t (short for a(t)), state as s_t, reward as r_t, target pose as p_t, command as c_t, and policy as $\pi_\theta(a_t|s_t, p_t, c_t)$, where θ represents the parameter of the neural network. Then the expected return can be written as:

$$J(\theta) = E_{\pi_\theta}(\sum_{t=0}^{T} \gamma^t r_t) \tag{1}$$

And the gradient of J_θ is:

$$\nabla_\theta J(\theta) = E_{\pi_\theta}\left[\left(\sum_{t=1}^{T} \nabla_\theta \log_{\pi_\theta}(a_t|s_t, p_t, c_t)\right)\left(\sum_{t=1}^{T} r_t\right)\right] \tag{2}$$

To reduce variance, causality can be exploited and a trick called "reward-to-go" can be applied:

$$\nabla_\theta J(\theta) = E_{\pi_\theta}\left[\left(\sum_{t=1}^{T} \nabla_\theta \log_{\pi_\theta}(a_t|s_t, p_t, c_t)\right)\left(\sum_{t'=t}^{T} r_{t'}\right)\right] \tag{3}$$

And with Monte-Carlo method, the above can be estimated as:

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} \nabla_\theta \log_{\pi_\theta}(a_{i,t}|s_{i,t}, p_{i,t}, c_{i,t})\left(\sum_{t'=t}^{T} r_{i,t'}\right) \tag{4}$$

where $i = 1, 2, \ldots, N$ represents the index of collected trajectories. Then, the parameters are updated as:

$$\theta \leftarrow \theta + \alpha \nabla_\theta J_\theta \tag{5}$$

2.1.2 Proximal Policy Optimization (PPO)

Though adopting some tricks to reduce the variance, VPG still suffers from serious instability in real applications. The intuitive reason for this is that VPG often take too large gradient steps, making it harder to converge.

To deal with this problem, PPO constrains the policy change in a small range [6]. Actually, PPO is an update version of Actor-Critic (AC), since their algorithm structures are exactly the same: both the old parameter θ_{old} and the new parameter θ are included. The only difference is that, in each iteration, θ is updated under constraints so that the new policy should not deviate too much from the old policy. After each iteration, the new parameter θ will turn into the old parameter θ_{old} of next iteration.

PPO's objective function is (for simplicity, take p_t and c_t as parts of s_t in this part):

$$J^{CLIP}(s_t, a_t, \theta, \theta_{old}) = E_{\pi_{\theta_{old}}}\left[min\big(d(s_t, a_t, \theta, \theta_{old})A_t, f(\epsilon, A_t)\big)\right] \tag{6}$$

$$A_t = \sum_{t'>t} \gamma^{t'-t} r_{t'} - V_{critic}(s_t) \quad (7)$$

And the parameter is updated as:

$$\theta \leftarrow \underset{\theta}{argmax}\ J^{CLIP}(s_t, a_t, \theta, \theta_{old}) \quad (8)$$

According to (7), $A_t = A(a_t, s_t, \theta_{old})$ measures how better an action is better than another action given (s_t, a_t, θ_{old}). $d(s_t, a_t, \theta, \theta_{old})$ parametrizes the divergence from the old policy and $f(\epsilon, A_t)$ is defined in (9):

$$d(s_t, a_t, \theta, \theta_{old}) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \quad (9)$$

$$f(\epsilon, A_t) = \begin{cases} (1+\epsilon)A_t, & A_t \geq 0 \\ (1-\epsilon)A_t, & A_t < 0 \end{cases} \quad (10)$$

where ϵ is a small pre-defined positive number.

With respect to AC's formulation, $\pi_\theta(a_t|s_t)$ is the probability that the new actor's policy do action a_t at state s_t, $\pi_{\theta_{old}}(a_t|s_t)$ is the probability that the old actor's policy do action a_t at state s_t, and $A = A(a_t, s_t, \theta_{old})$ is the advantage estimated based on the old critic.

Intuitively, when $A_t \geq 0$, if π_θ can be very big, VPG will take a huge gradient step which should be avoided, while PPO constrains the step size to be at most $(1+\epsilon)A_t$. The same logic applies to $A_t < 0$. Consequently, the new policy will never deviate too much from the old policy, ensuring the stability of PPO algorithm. Also for this reason, PPO is the most popular RL algorithm currently and proves to perform best in this work.

3 Model

3.1 Model Structure

Based on the discussion on overall model structure and RL algorithms, the general model is designed as Fig. 1. Details are neglected for simplicity. The below model is based on PPO, while VPG model will be discussed in ablation study. The model mainly include 3 parts: rollout generator, rollout sampler, and policy trainer.

In the first part, the action is calculated with policy π^A (represented as neural network), which then generated rewards and observation based on the physics model defined by the robot and the environment. The rewards are calculated through assessing the similarity between the robot's real configuration and dynamics with the motion clips and commands given. The above process repeats until enough rollouts haven been generated.

In the second part, rollouts are randomly sampled from the buffer (containing previous rollouts), and with PPO, clip loss is calculated for both actor and critic. Intuitively, actor loss is the loss function that depicts the loss of current policy, while critic loss depicts how good the value evaluation is.

In the third part, the loss is used to train the policy, where the critic is first trained because the training of policy π^A requires the latest version of π^C to improve training efficiency.

The above three parts consist of an iteration, and the number of iterations is predefined as 30000. The training will cease if the number of iterations have been reached or the model performance is good enough (judging from the change in total reward). After one training, the hyper-parameters will be tuned for next training.

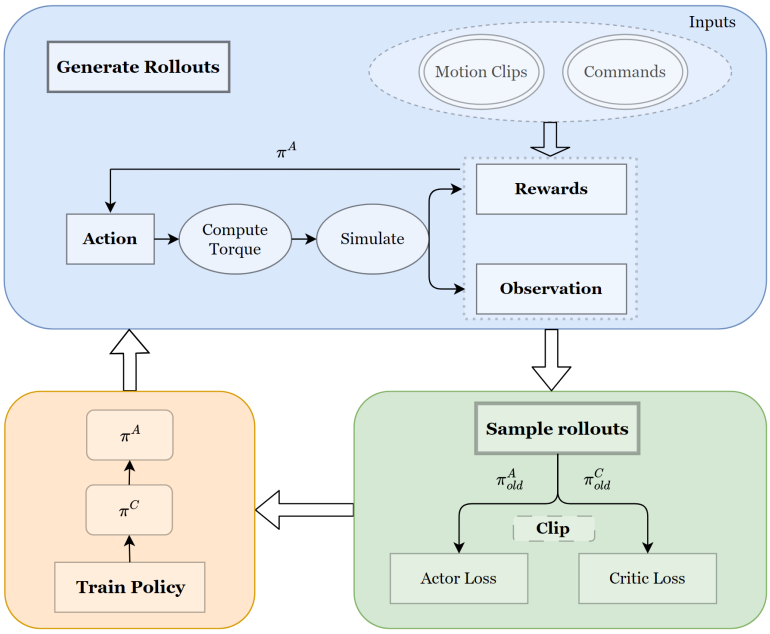

Fig 1. Model structure

3.2 Policy Representation

3.2.1 Observation (State)

Although this is not a partial-observable problem, but the framework can be straightforwardly applied to problems with partial-observability, thus the word 'state' is substituted with 'observation' here while maintaining the meaning.

The observation describes the robot's whole configuration, including each joint's relative position to the base link, rotation, linear and angular velocity. All the features are calculated in the local coordinate frame, with the base link as the origin and the robot's facing direction as the x-axis. For arrays of target poses $\{p_t\}$ that expressed in the space frame, they are translated to local frame through linear transformation. The similar trick is applied to the commands $\{c_t\}$.

3.2.2 Action

The action specifies each PD controllers' target angles. For spherical joints (e.g. shoulders), the action is represented in axis-angle form. For revolute joints, the action is represented in scalar form, specifying the target rotation angle. The uses of action as the desired value of PD controllers can avoid low-level control that involve torques, leading to better performance [7].

3.2.3 Reward

The reward setting is the core of this work. The idea is simple: through the setting of reward, the robot should be encouraged to walk steadily. Since there are reference clips available, this question can be split into two parts: on one hand, the robot should imitate the clip so as to be able to walk; on the other hand, the robot should be able to robustly walk in all directions given commands. So, the reward at time t can be thus split into two parts:

$$r_t = \omega^I r_t^I + \omega^C r_t^C \tag{11}$$

The I refers to 'Imitation' and C refers to 'Commands'. ω^I and ω^C differs as $\{c_t\}$ changes. r_t^I should reflect the similarity between the target pose and real pose, so refer to DeepMimic, r_t^I can be written as:

$$r_t^I = \omega^{JP} r_t^{JP} + \omega^{JV} r_t^{JV} + \omega^{EP} r_t^{EP} + \omega^{RP} r_t^{RP} + \omega^{RV} r_t^{RV} \tag{12}$$

which is specified in Table 1. All the parameters are obtained from trail-and-error.

Table 1. Imitation reward setting

	Physical meaning	Expression	Scale ω^*
r_t^{JP}	Measuring the similarity of joint position	$\exp\big(-2\sum_i(\hat{q}_{j,t}^i - q_{j,t}^i)^2\big)$	0.5
r_t^{JV}	Measuring the similarity of joint velocity	$\exp\big(-0.1\sum_i(\hat{v}_{j,t}^i - v_{j,t}^i)^2\big)$	0.05
r_t^{EP}	Measuring the similarity of end-effector position	$\exp\big(-40\sum_i(\hat{q}_{e,t}^i - q_{e,t}^i)^2\big)$	0.2
r_t^{RP}	Measuring the similarity of root position	$\exp\big(-20\sum_i(\hat{q}_{r,t}^i - q_{r,t}^i)^2\big)$	0.15
r_t^{RV}	Measuring the similarity of root velocity	$\exp\big(-2\sum_i(\hat{v}_{r,t}^i - v_{r,t}^i)^2\big)$	0.1

* *i refers to the index of joints, $\hat{x}$ refers to the target configuration, x refers to the real configuration, ω^* refers to ω^{JP}, ω^{JV}, etc. All the variables are in vector form.*

Similarly, r_t^C is defined as:

$$r_t^C = \exp\big(-2\sum_i(\hat{v}_{e,t}^i - v_{e,t}^i)^2\big) \quad (13)$$

3.2.4 Policy

As is discussed above, the policy $\pi(a_t|s_t,p_t,c_t)$ maps (s_t,p_t,c_t) to action a_t. The mapping takes form in neural network. Since the action space is continuous, the output is modelled as a Gaussian distribution with a fixed covariance. Thus, the output of the neural network is simply the mean of the distribution, with the dimension equal to total degrees of freedom. In this work, the policy is queried at 60Hz.

$$\pi(a_t|s_t,p_t,c_t) = N(\mu(s_t,p_t,c_t),\Sigma) \quad (14)$$

The detailed network structure is shown in Fig. 2.

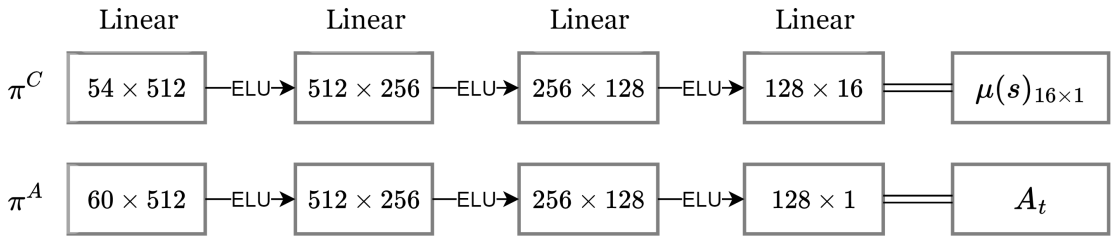

Fig 2. Network structure of PPO (consisting of actor and critic)

4 Experiments

In this section, we first discuss experimental settings in Sec.4.1, then introduce the training setting and process in Sec.4.2, and discuss the results in Sec.4.3. In Sec.4.4, ablation study with VPG is carried out.

4.1 Environment Settings

4.1.1 Robot Description

The humanoid robot model in this work is from ISRLab, shown in Fig. 3. There are 16 joints in all, including arm, shoulder, elbow, hand, knee, leg and etc. The detail description are defined in universal robot description file (URDF), which will not be discussed here. The controller type is P (Proportional) controller, and the stiffness and damping are respectively defined for all joints.

In the simulation, many random noises have been added to guarantee the robustness of training. The friction coefficients between joints and links are randomized between 0.25 and 1.75, and the mass of base and legs are fluctuating in 5% level. Periodical pushes are imposed on the robot with an interval of 5s and maximum pushing force 0.3N.

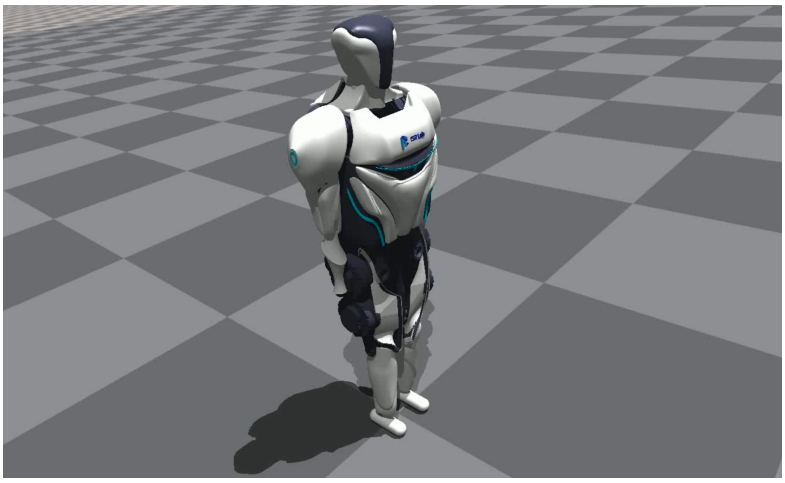

Fig 3. ISRLab Humanoid Robot

4.1.2 Gym Setting

The simulation environment is based on Isaac Gym by NVIDIA [8]. The advantage of Isaac Gym is that all the variables are stored as GPU tensors and all the computations are carried out in GPU. This can significantly accelerate training process and enable parallel computing.

The environment type is set as rough plane. The static and dynamic friction coefficient is set as 1.0, and there are random height changes in the terrain, ranging from 0.05m to 0.05m. The initial state sets all joint angles and velocities to 0, while the root position of robot is chosen randomly after each resetting.

The minimum simulation time-step is set as $\Delta t = 0.01s$, and gravity is set as $\vec{g} = [0, 0, -9.81]$. The computation of contact force between the robot and the gym is very complex, and is set by default.

4.1.3 Reference Clips Data

The reference clips used in this work are saved as .txt and loaded as np.array. In essence, these clips are series of coordinates of all the 16 joints that are recorded when the robot is walking steadily. To obtain the reference clips data, the robot is manipulated by mouse in a simulation environment to reach all the poses in a walking cycle.

4.2 Training

The optimizer uses Adam, with initial learning rate 1×10^{-3}. The ϵ in (6) is set to be 0.2. The decaying factor γ of rewards in (7) is set to be 0.998. The ω^I and ω^C in (11) are obtained by line search under constraint: $\omega^I + \omega^C = 1$. The training lasts for around 10k iterations, and in each iteration, about 500 rollouts are generated first and then sampled to update the policy. The loss curve are shown in Fig. 4. As is discussed in Section 2, the loss can be split into actor loss and critic loss. Since the loss is extremely volatile, the curves have been smoothed. It's clear that both loss become very small at the end of training.

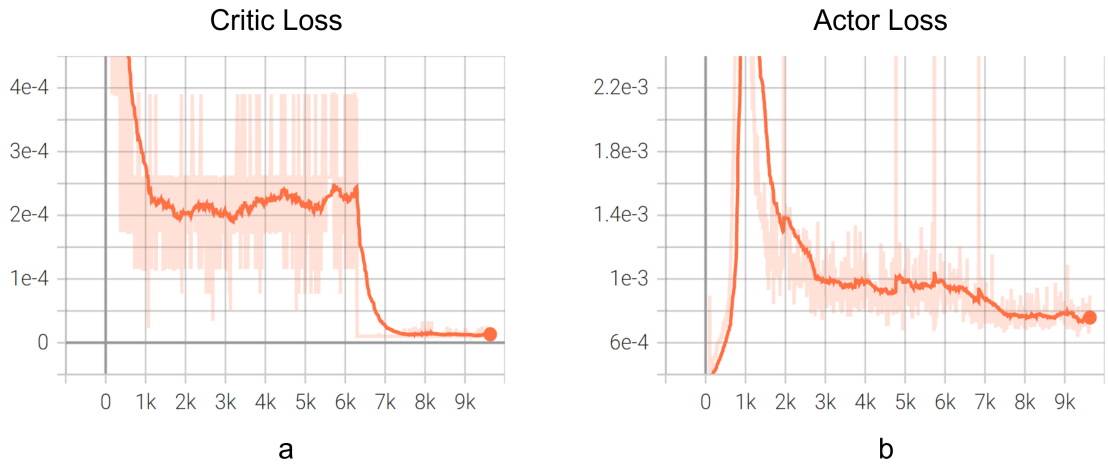

Fig 4. (a)Critic loss curve. (b)Actor loss curve. For both curves, the lighter one refers to the actual curve and the darker one refers to the smoother curve.

However, the mean reward curve in Fig. 5 does not match the loss curve in Fig. 4. The maximum of mean rewards takes value around 6k iterations, which means that the robot behave the best at around 6k iteration. The reason behind this contradiction is that, the loss in PPO does not reflect the 'real' loss, but the difference between past estimations and current estimations. The PPO problem is in essence a fixed point iteration which will converge at last theoretically, so the loss has no real physical meanings, unlike classical ML problems. In this sense, reward is a more essential metric. So, for Section 4.3, the result given by the model at 6k iteration will be discussed.

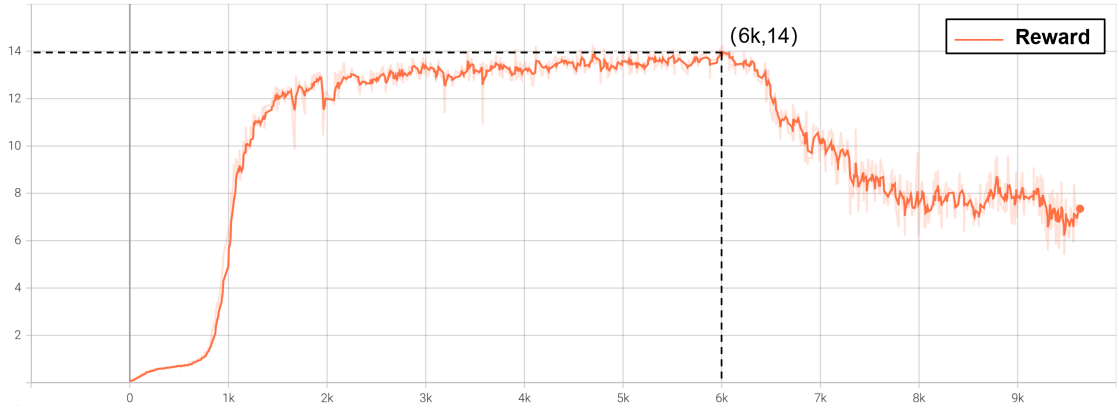

Fig 5. Mean Reward Curve

4.3 Result

Using the model at 6000^{th} iteration, an animation is generated. As shown in Fig. 6, the robot can walk steadily. Let the original direction that the robot is facing be $x = [1, 0, 0]$, the command given is $c(t) = [\sqrt{3}/2, 1/2, 0]$. It is clear that the robot succeeds in turning around to the direction given by $c(t)$ while keeps walking steadily.

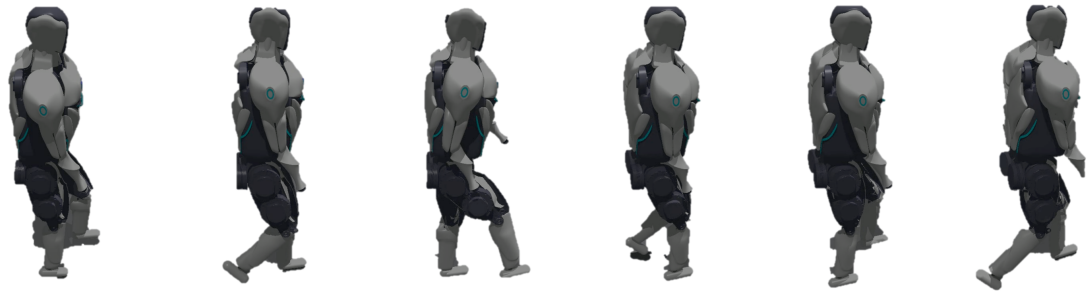

Fig 6. Robot's walking in one cycle (from moving left leg to settling left leg). For full video, see the attachment *walk.mp4*.

To see clearly how the robot walks, Fig. 7 shows the positions and torques of all the 16 joints. Ideally, all the positions and torques should be periodical since walking is a periodical movement. According to Fig. 7(a), some of the original stages of positions do not show good periodicity because the robot is still turning, which means that some of the joints have to make periodical movements. In later stages, nearly all joints move in a stable way, as can also be seen in Fig. 7(b). Joints' positions are torques are one-to-one with a fixed delay caused by the P controller.

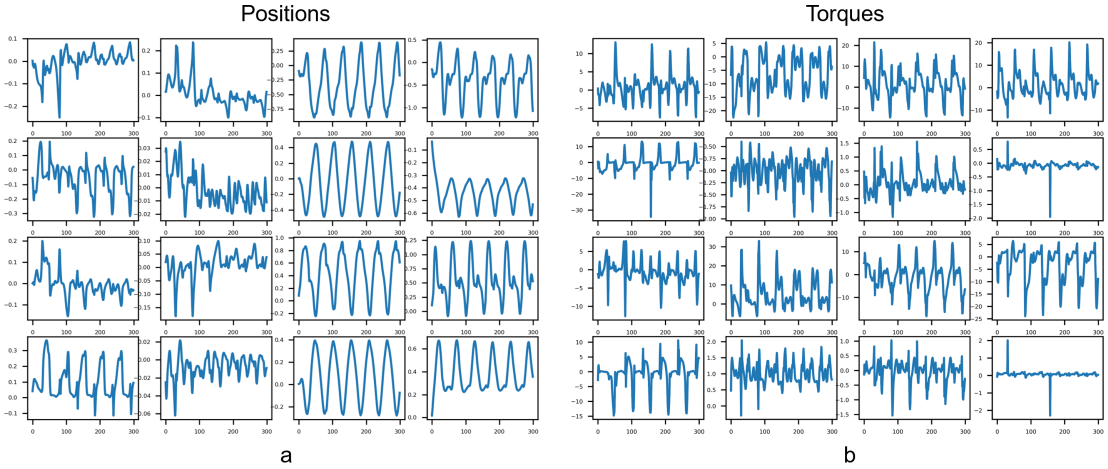

Fig 7. The positions and torques of all 16 joints of the robot. The x-axis refers to the index of time-step, the y-axis of positions refers to the percentage of shift (compared to shift limit), and the y-axis of torques has a unit of $N \cdot m$.

4.4 Ablation Study

For ablation study, VPG is adopted in comparison with PPO. According to Section 2.1.1, the policy π_θ of VPG is also represented by a neural network, the structure of network is shown in Fig. 8. For comparison, the structure is chosen to be exactly the same with the actor's network structure of PPO. The parameters tuning is the same with Section 4.2.

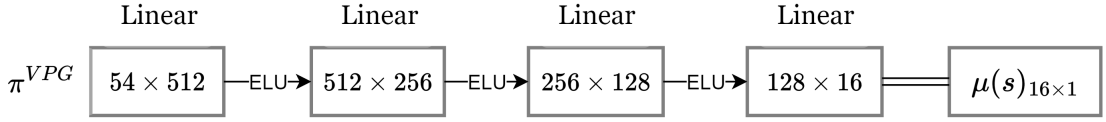

Fig 8. Network Structure of VPG for Ablation Study

Besides the RL algorithm chosen, the reward setting is also taken into consideration for ablation study to prove that it is very crucial. Another setting is chosen to be:

$$r'_t = \begin{cases} 1, & \textit{Not fallen} \\ 0, & \textit{Fallen} \end{cases} \tag{15}$$

where 'fallen' means that any part other than feet of the robot has contacted with the ground.

The result is shown in Table 2. Only with PPO and sophisticated reward setting (r_t) can the robot walk steadily. Without a careful design of reward, for example simply adopting r'_t in (15), the robot cannot even walk. With r_t, the robot can walk for a few steps then fall.

Table 2. Ablation Study Result

Algorithm	Reward	Walking?
VPG	r_t	✓ (Only for few steps)
VPG	r'_t	✗
PPO	r_t	✓
PPO	r'_t	✗

From above discussion, it can be proved that reward setting is very important in RL task, and r_t defined in this work is very successful in depicting the goal for the robot. It can also be proved that PPO is superior to VPG in that it can find a better local minimum so as to keep the robot walking steadily, while VPG gets stuck very early in a bad policy.

Consequently, the ablation study proves that both PPO and the design of r_t are very crucial in this work.

5 Conclusions

This paper mainly refers to DeepMimic and deals with the problem of training humanoid robot to walk steadily. To exploit the reference clips, I carefully design a reward function that can be split into two parts, where one part measure the similarity between real poses and target poses and another part measures how well the robot answers the command. Under a standard RL framework, PPO is chosen to be the base algorithm. Through training, the robot succeeds in walking steadily and smartly, which is able to turn its direction according to the command given. This work further proves that data-driven RL method is very promising in RL tasks that are increasingly complicated.

6 Reference

[1] Perttu Hämäläinen, Joose Rajamäki, and C Karen Liu. 2015. Online control of simulated humanoids using particle belief propagation. ACM Transactions on Graphics (TOG) 34, 4 (2015), 81.

[2] Yoonsang Lee, Moon Seok Park, Taesoo Kwon, and Jehee Lee. 2014. Locomotion Control for Many-muscle Humanoids. ACM Trans. Graph. 33, 6, Article 218 (Nov. 2014), 11 pages.

[3] Sergey Levine, Jack M. Wang, Alexis Haraux, Zoran Popović, and Vladlen Koltun. 2012. Continuous Character Control with Low-Dimensional Embeddings. ACM Transactions on Graphics 31, 4 (2012), 28.

[4] Jonathan Ho and Stefano Ermon. 2016. Generative Adversarial Imitation Learning. In Advances in Neural Information Processing Systems 29. Curran Associates, Inc., 4565–4573.

[5] Xue Bin Peng, Pieter Abbeel, Sergey Levine and Michiel van de Panne. 2018. DeepMimic: Example-Guided Deep Reinforcement Learning of Physics-Based Character Skills. ACM Transactions on Graphics 37, 4(2018), 14.

[6] John Schulman, Filip Wolski, Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. https://arxiv.org/abs/1707.06347

[7] Xue Bin Peng, Glen Berseth, KangKang Yin, and Michiel van de Panne. 2017a. DeepLoco: Dynamic Locomotion Skills Using Hierarchical Deep Reinforcement Learning. ACM Transactions on Graphics (Proc. SIGGRAPH 2017) 36, 4 (2017).

[8] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, Gavriel State. 2021. Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning. https://arxiv.org/abs/2108.10470