ECE 239AS: Final Project Presentation

Chenyang Wang, Gongjie Qi, Hanqing Wu, Xiao Zeng

Department of Electrical & Computer Engineering, UCLA

INTRODUCTION

In this project, we study the paper *DeepMimic: Example-Guided Deep Reinforcement Learning of Physics-Based Character Skills*, Peng et al, 2018. In particular, we follow the steps described in the paper where we first synthesize a motion controller that takes a character model, a corresponding set of kinematic reference motions and a reward function representing a specific task as inputs. We then use neural networks with proximal policy optimization (PPO) algorithm to compute a control policy that enables the character to imitate the reference motions and achieve the task goal.

We find this project particularly interesting because it tries to use reinforcement learning techniques in the field of modeling the motions of humans and animals. Currently, manually designed controllers show satisfying results in reproducing certain motions, however a main problem here is that these controllers are not generalizable, i.e. they cannot handle new skills and new situations correctly. Another major obstacle of current physics-based human and animal motion simulation approaches is the directability. It is still extremely challenging for users to deliberately elicit some desirable behaviors from simulated characters. Therefore, it is really interesting to see how reinforcement learning concepts can be used to solve these kinds of problems. What's more, motion modeling techniques are applied in many fields from biomechanics to robotics and animation, which are among the hottest research areas recently.

OBJECTIVES

In this project our main objectives are reproducing the results demonstrated in the DeepMimic paper and applying the model to a new environment. More specifically, we train the models using the original hyper-parameters and humanoid model mentioned on the paper to reproduce results. We then compare the results of ours and the paper and find areas that can be improved. To implement the model in a new environment, we apply the framework to a modified 3D simulated character other than the humanoid model widely demonstrated in the paper. We show that the DeepMimic model also produces a robust result in the new environment.

METHODS

To begin with, as described in the paper [1], the whole system can be explained as receiving a character model as input, together with a corresponding set of kinematic reference motions and a task defined by a reward function. The system then synthesizes a controller that enables the character to imitate the reference motions while satisfying the task objectives. Each reference motion is represented as sequence of target poses. A control policy then maps the state of the character, to a task-specific goal to an action, which is then used to compute torques to be applied to each of the character's joints. Proportional-Derivative (PD) controllers are used together with target angles as inputs to produce the final torques applied at the joints. The reference motions are used to define an imitation reward and the goal defines a task-specific reward. The final result of the system is a policy that enables the simulated character to imitate the behaviours from the reference motions while fulfilling the specified task objectives. The policies are modeled using neural networks and trained using the proximal policy optimization algorithm, which will discussed in the following.

The **state** is used to describe the configuration of the character's body. The features are the relative positions of each link with respect to the root, their rotations expressed in quaternions, and their linear and angular velocities. Note that all features are computed in the character's local coordinate frames with the root at the origin and the x-axis along the root link's facing direction. The **action** from the policy specifies target orientations for PD controllers at each joint. Note that the policy is queried at 30Hz and target orientations for spherical joints are represented in axis-angle form and targets for revolute joints are represented by scalar rotation angles.

In terms of the Neural Network, each policy is represented by a neural network that maps a given state and goal to a distribution over action. The action distribution is modeled as Gaussian. The inputs are feeded to two fully connected layers with 1024 and 512 units each, followed by a linear output layer. ReLu is the activation function used for all layers. The value function is modeled by a similar network without the output layer.

In terms of the learning algorithm, as the authors pointed out, in practice the policy gradient estimator suffers from high variance and is unstable during learning. In order to deal with problem, Trust Region Policy Optimization (TRPO) is proposed in 2015 by Schulman et al. TRPO optimizes the same objective while includes an additional KL-divergence constraint to prevent the current policy from deviating too far from the previous policy. In this paper, Proximal Policy Optimization (PPO) [2] is used instead of TRPO. PPO replaces the hard constraint by optimizing a surrogate loss. In particular, clipped PPO is used in this paper. The algorithm is presented below:

ALGORITHM 1: Proximal Policy Optimization

```
θ ← random weights
ψ ← random weights
while not done do
    $s_0$ ← sample initial state from reference motion
    Initialize character to state $s_0$
    for step = 1, …, m do
        s ← start state
        $a \sim \pi_\theta(a|s)$
        Apply a and simulate forward one step
        s′ ← end state
        r ← reward
        record (s, a, r, s′) into memory D
    end for

    $\theta_{old} \leftarrow \theta$
    for each update step do
        Sample minibatch of n samples $\{(s_i, a_i, r_i, s'_i)\}$ from D

        Update value function:
        for each $(s_i, a_i, r_i, s'_i)$ do
            $y_i$ ← compute target values using TD(λ)
        end for
        $\psi \leftarrow \psi + \alpha_v \left(\frac{1}{n}\sum_i \nabla_\psi V_\psi(s_i)(y_i - V(s_i))\right)$

        Update policy:
        for each $(s_i, a_i, r_i, s'_i)$ do
            $\mathcal{A}_i$ ← compute advantage using $V_\psi$ and GAE
            $w_i(\theta) \leftarrow \frac{\pi_\theta(a_i|s_i)}{\pi_{\theta_{old}}(a_i|s_i)}$
        end for
        $\theta \leftarrow$
        $\theta + \alpha_\pi \frac{1}{n}\sum_i \nabla_\theta \min(w_i(\theta)\mathcal{A}_i, \mathrm{clip}(w_i(\theta), 1-\epsilon, 1+\epsilon)\mathcal{A}_i)$
    end for
end while
```

In particular the policy updates after a batch of 4096 samples has been collected, and then a minibatch of size 256 are sampled at each gradient step. The discount factor is 0.95. λ = 0.95 is used for both TD(λ) and GAE(λ). The likelihood ratio clipping threshold is 0.2, and value function step size is 10e-2, policy step size is 5*10e-5.

RESULTS

To reproduce the results of DeepMimic model and ensure that we are in good standing to further modify and possibly optimize the framework, we trained the model on walking and running tasks using the original hyper-parameters and humanoid character. We used Microsoft Visual Studio 2017 was chosen as the running environment and 12 worker processes during training to fully utilize CPU cores. The resulting policies of walking and running are satisfying, in the sense that not only the humanoid character can walk and run like a normal human but also the value functions have high and stable outputs. We saved both intermediate and final results during the training of running task and they are shown in Figure 1 and Figure 2.

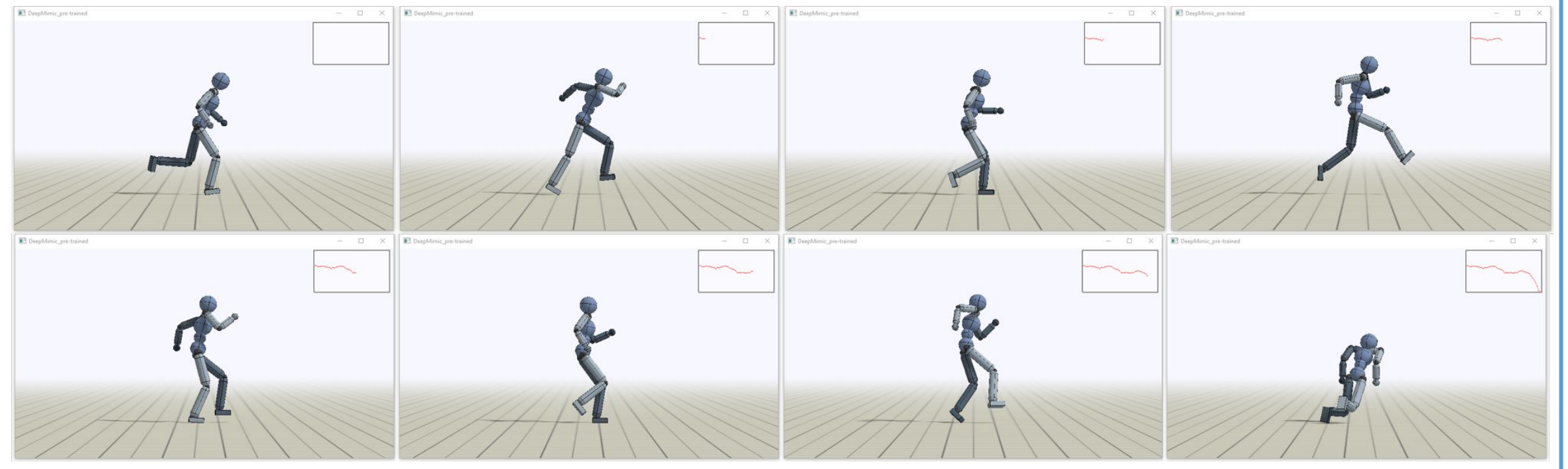

Figure 1. From left to right, top to bottom: simulated humanoid runs based on an intermediate policy trained on 2.2 million samples. Plots on the upper-right corner present the prediction of value function.

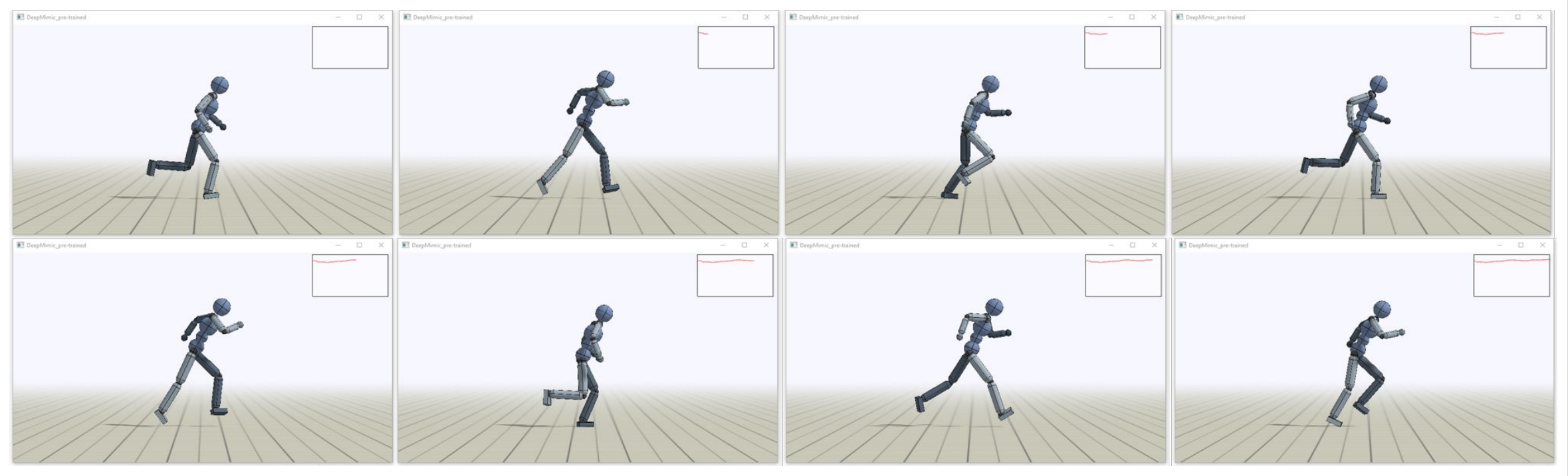

Figure 2. From left to right, top to bottom: simulated humanoid runs based on the resulting policy training on 60.1 million samples. Plots on the upper-right corner present the prediction of value function.

Figure 1 depicts the snapshots of the humanoid running based on a policy trained using 2.2 million samples after 500 iterations. We can see the humanoid easily falls down due to imbalance gait and the plotted outputs of value function dramatically drops to zero, which indicates the episode is failed. Figure 2 illustrates the humanoid running using the output policy trained on 60.1 million samples after 11038 iterations, which took about 22 hours. The gait of the simulated character are more balanced than that of the intermediate result. The value function outputs are also both higher and more stable than those shown in Figure 1. This means that we successfully reproduce the results of the DeepMimic model.

We then trained the model using a modified humanoid character on the running task to explore further explore its performance when be applied to a different environment. The left hip component of the new humanoid has a mass 10 times large as that of the original humanoid to simulate a character with heavily imbalanced body and same configuration was used for training. The obtained policy after training for 8406 iterations and about 16 hours is surprisingly good. The snapshots of intermediate and resulting policy are presented in Figure 3 and 4. Outputs of value functions are also shown in the upper-right corners of the snapshots.

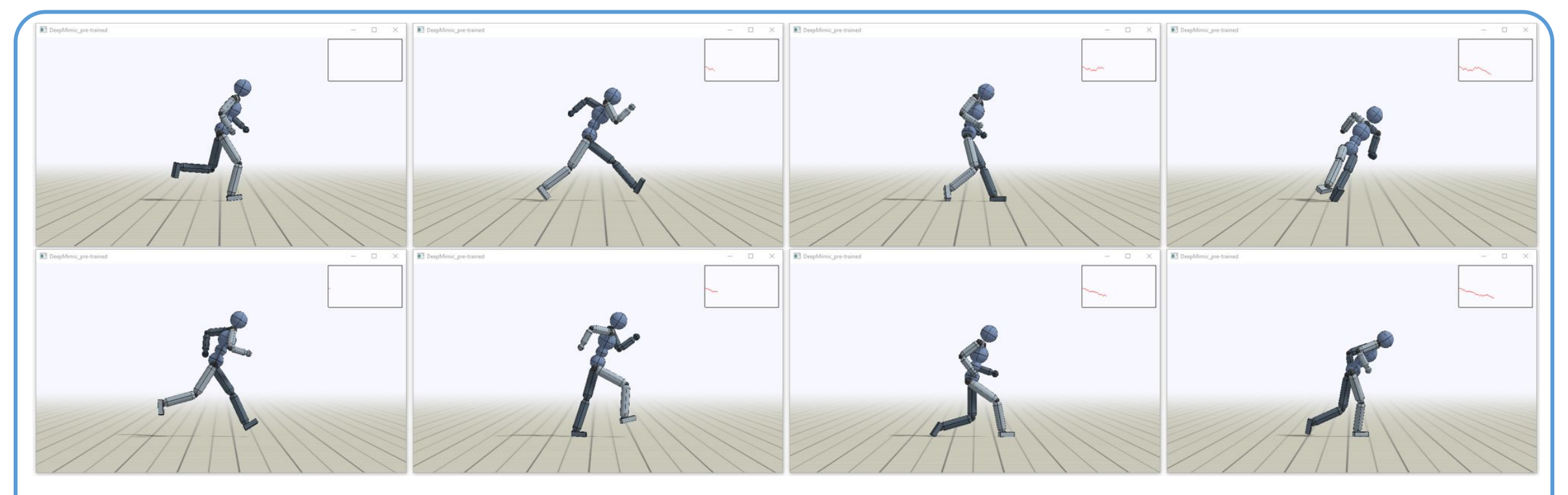

Figure 3. From left to right, top to bottom: modified humanoid with larger left hip mass runs based on an intermediate policy trained on 1.3 million samples. Plots on the upper-right corner present the prediction of value function.

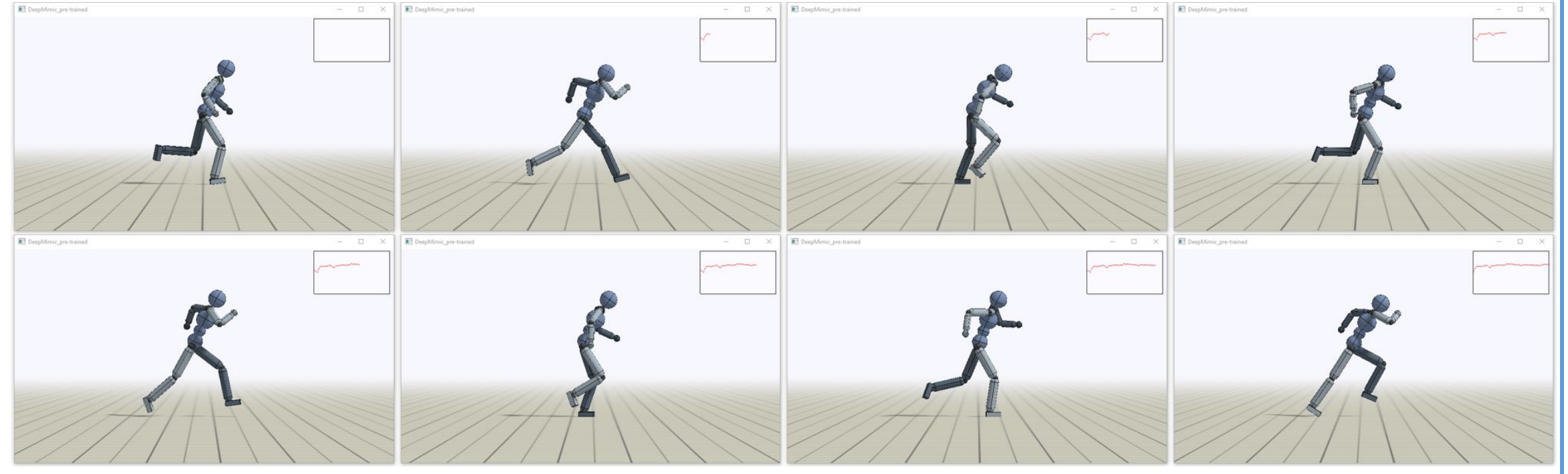

Figure 4. From left to right, top to bottom: modified humanoid with larger left hip mass runs based on the resulting policy trained on 40.3 million samples. Plots on the upper-right corner present the prediction of value function.

We can clearly see from the above figures that although initially the character easily falls down to its heavier side, it learns a policy through the training process to overcome its imbalanced mass and can run with some tolerably swings. It is also noticeable that the resulting value functions have both lower and less stable outputs in the imbalanced character case than that of the original humanoid.

CONCLUSIONS

In this project we studied and implemented a data-driven deep reinforcement learning framework for training control policies for simulated characters. By implementing the model in both original and new environments, we managed to reproduce acceptable results including different motion tasks as well as different mass distribution in the character model. Our results proves this deep RL framework's robustness and capability of producing natural motions without perturbations and nearly indistinguishable from their original motion capture data.

DISCUSSION

Our experiments on the model proves the robustness and capabilities of this model to perform basic tasks. As for the future work, we wish to integrate more complicated motion tasks for the character models, more complex tasks that involve not only self motions, but also interaction between the environment objects. What is more, the training period will be shortened if we have more knowledge on control strategies so that the controllers are set in more optimized ways.

REFERENCE

[1] Peng, X.B., Abbeel,P., Levine,S. & Panne, M. (2018) DeepMimic: Example-Guided Deep Reinforcement Learning of Physics-Based Character Skills. ACM Trans.

[2] Schulman, J., Wolski, F., Dhariwal,P., Radford,A. & Klimov, O. (2017) Proximal Policy Optimization Algorithms. OpenAI.