EE239 Reinforcement Learning Project Final Report

Chenyang Wang
Department of ECE
UCLA
105425757

Gongjie Qi
Department of ECE
UCLA
805429380

Hanqing Wu
Department of ECE
UCLA
005429548

Xiao Zeng
Department of CS
UCLA
505030377

Abstract

In this report we study the algorithms proposed in the paper DeepMinic by Peng et al. [1] We reproduce the results of the paper and demonstrate different results obtained when we alter the mass distribution of the character model or use different motion capture data.

1 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.

2 Main Objectives

Our main objectives in this project 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 new environments, we apply the framework to a modified 3D simulated character other than the humanoid model widely demonstrated in the paper as well as a new motion capture data. We show that the DeepMimic model is capable of producing a robust result in the new environment as long as its reference data contains a complete cycle of actions.

Preprint. Under review.

ALGORITHM 1: Proximal Policy Optimization

$\theta \leftarrow$ random weights
$\psi \leftarrow$ random weights
while not done **do**
  $s_0 \leftarrow$ sample initial state from reference motion
  Initialize character to state s_0
  for step = 1, …, m **do**
    $s \leftarrow$ start state
    $a \sim \pi_\theta(a|s)$
    Apply a and simulate forward one step
    $s' \leftarrow$ end state
    $r \leftarrow$ 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**
      $y_i \leftarrow$ 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**
      $\mathcal{A}_i \leftarrow$ compute advantage using V_ψ and GAE
      $w_i(\theta) \leftarrow \frac{\pi_\theta(a_i|s_i)}{\pi_{\theta_{old}}(a_i|s_i)}$
    end for
    $\theta \leftarrow$
    $\theta + \alpha_\pi \frac{1}{n} \sum_i \nabla_\theta \min\left(w_i(\theta)\mathcal{A}_i, \text{clip}\left(w_i(\theta), 1-\epsilon, 1+\epsilon\right)\mathcal{A}_i\right)$
  end for
end while

Figure 1: The pseudo-code for Proximal Policy Optimization algorithm

3 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, 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 in Algorithm 1 as shown in [1]. 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. $\lambda = 0.95$ 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.

4 Results

4.1 Walking and Running

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 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 2 and Figure 3.

Figure 2 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 3 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 2. This means that we successfully reproduce the results of the DeepMimic model.

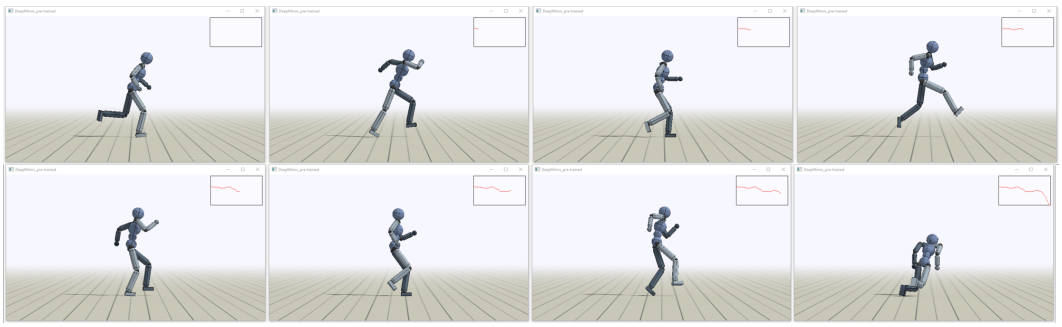

Figure 2: From left to right, top to bottom: snapshots of simulated humanoid runs based on the intermediate policies trained on 2.2 million samples after 500 iterations. The plots on the upper-right corner of snapshots present the prediction of value function.

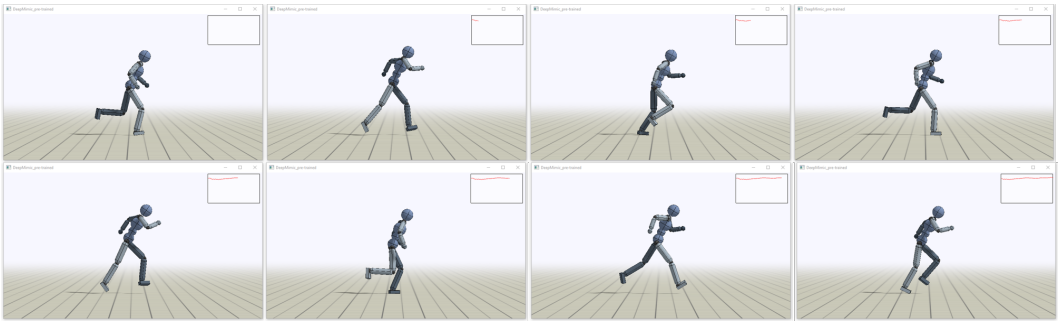

Figure 3: From left to right, top to bottom: snapshots of simulated humanoid runs based on the intermediate policies training on 60.1 million samples after 11038 iterations. The plots on the upper-right corner of snapshots present the prediction of value function.

4.2 Character model with altered mass distribution

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 4 and 5. Outputs of value functions are also shown in the upper-right corners of the snapshots.

In Figure 4, snapshots of the the humanoid with larger left hip mass running based on a policy trained using 1.3 million samples after 300 iterations are shown. We can clearly see from the pictures that the character easily falls down in due to its heavier side and the value function outputs drops quickly. Snapshots of the character running base on a policy trained using 40.3 million samples after 8406 iterations are presented in Figure 5. The modified humanoid learns a policy through the training process to overcome its imbalanced mass and can run with some tolerably lagging on its left leg. 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. Figure 6 compares the gait between the original and the altered character from a front view at same frames during the simulation. The humanoid with a heavier left hip shown on the upper snapshots can normally run with swings more towards its right side while the original humanoid shown in the lower snapshots sways to left and right evenly. This means the DeepMimic framework is robust enough that the trained character can automatically develop a way to perform tasks like running to adapt a new environment such as a different mass distribution of the simulated character.

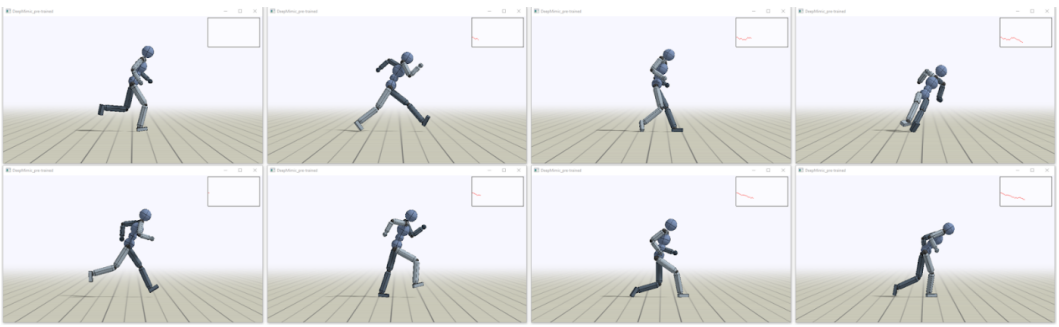

Figure 4: From left to right, top to bottom: modified humanoid with larger left hip mass runs based on an intermediate policy trained on 1.3 million samples after 300 iterations. Plots on the upper-right corner present the prediction of value function.

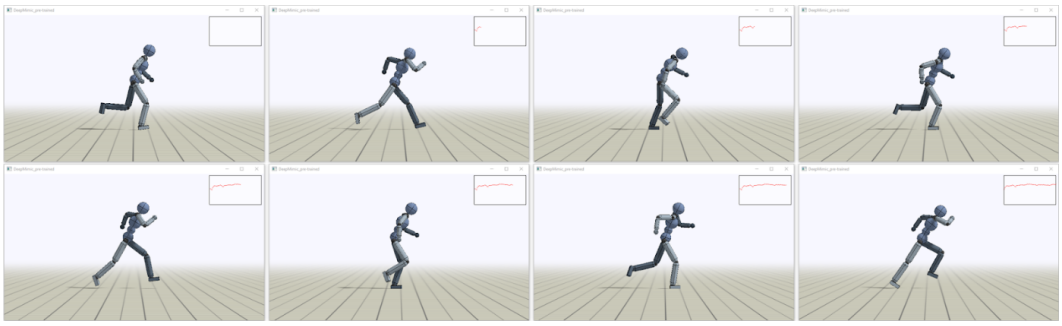

Figure 5: From left to right, top to bottom: modified humanoid with larger left hip mass runs based on the resulting policy trained on 40.3 million samples after 8406 iterations. Plots on the upper-right corner present the prediction of value function.

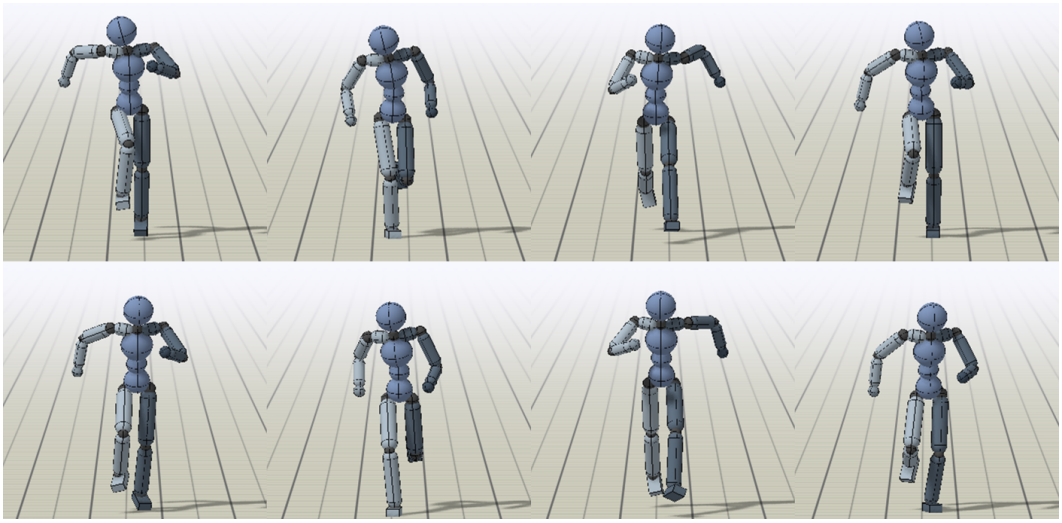

Figure 6: Top: Front view of the altered humanoid humanoid running. Bottom: Front view of the original humanoid running at the same frames as the altered humanoid during simulation.

4.3 Other motion capture data

We also tried to study the adaptability of the model by training it using another less well-captured online motion data. Experiments were conducted using the unmodified humanoid model with balanced mass distribution. The new motion data [4] captures a jump hop and was converted to a required JSON format containing frame and rotation information of various body parts. However, it is different from the original reference data used by the author because it does not fully capture a complete cycle of the jump hop action. More specifically, a part of the actions between the character's landing and its preparation for the next jump are missing in the data. This makes the new motion data similar to that captured in the wild and thus very interesting to be applied to training the model.

We closely monitored the intermediate policy when running the experiment. After training the model on 30 million samples for 6870 iterations, we found that although the policy allows the simulated character to stay balanced and the value function to have stable returns, the simulation looks very different from the motion capture data. A comparison of the motion capture data and the simulated actions is shown in Figure 7. The humanoid learns a policy that outputs similar joint orientations as the reference motion but fails to jump.

We think this can be caused by the incomplete motion cycle contained in the reference data, which stops the character from learning a set of consistent actions from preparation of jumping to landing. The overemphasis on matching joint orientations over matching center-of-mass of the reference data during training can be another cause to the failure of correct imitation in this experiment. Thus

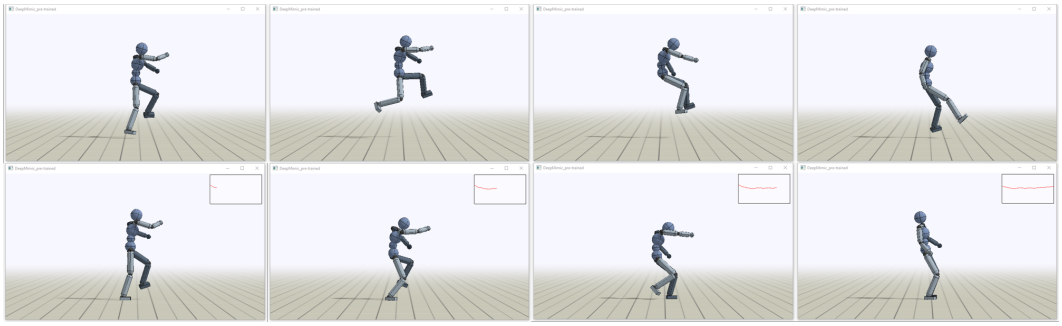

Figure 7: Top: snapshots of the character performs jump hop based on the motion capture data. Bottom: snapshots of the character performs jump hop based on the policy trained on 30 million samples after 6870 iterations, at the same frames as the motion capture data during simulation.

Motion	Number of Samples (10^6)	Normalized Return
Run using original humanoid	60	0.915
Run using altered humanoid	40	0.783
Jump hop using original humanoid	30	0.620

Table 1: Performance statistics of motion imitation measured by normalized test return over 600 timesteps in one episode, with 1 being the maximum return, and 0 the minimum return.

it seems that the current DeepMimic model still requires well-captured and complete reference motion data to adequately learn an action. It might also need weight adjustments for imitating joint orientations and center-of-mass when be trained on motions with significant vertical shift of body.

The model performances are also measured using normalized test return over 600 timesteps in one episode for each of our attempts and are illustrated in Table 1. Training the humanoid to run using the original character, which is our first attempt, produces very high average return. The normalized return of the policy trained using altered character is also satisfying given its smaller number of training samples than that of the first attempt. Note that in our third attempt, although the character fails to mimic the jump action when training using the jump hop motion capture data, the average return per episode remains relatively robust.

5 Conclusion and Discussion

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 that catches a complete cycle of actions.

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.

In this project, all team members carefully studied the DeepMimic paper and model. Chenyang Wang mainly contributed to the writing of introduction and method sections in the poster and reports. Gongjie Qi worked on the writing of introduction, conclusion and discusiion sections. Hanqing Wu contributed to the presentation as well as some sections in reports. Xiao Zeng took charge of running the experiments and report results in the results section.

References

[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*.

[3] Duan, Y., Chen, X.,Houthooft,R.,Schulman,J. & Abbeel, P. (2016) Benchmarking Deep Reinforcement Learning for Continuous Control. *OpenAI*.

[4] Mocap Data. https://github.com/Zju-George/DeepMimic/blob/master/data/motions/HMMR2DM/jump_hop.txt