# ECE239AS Course Project
# An Application of RL on Imitating Motionclips

Bohan Chen [1] Parth Agrawal [2] Matthew Ruiz [2] Arihant Jain[2]

[1]Department of Mathematics [2]Department of Computer Science

## Introduction

Our course project is based on the paper [1], where Peng et al. [1] applied reinforcement learning (RL) methods to learn robust character control policies. Their learning framework provides an approach to multiple tasks:

- Imitating a broad range of example motionclips.
- Learning complex recoveries.
- Adapting a certain motion to several different morphologies.
- Accomplishing user-specified goals.

There are two kinds of objective in their RL framework, namely, the motion-imitation objective and the task objective. Combining these two objectives, the trained characters can react intelligently in interactive settings, e.g., by walking in a desired direction or throwing a ball at a user-specified target.

Generally speaking, RL approaches are applied to seek optimal strategies to maximize long-term rewards. In some game-based models, such as chess or Go, the rewards are corresponding to the final result, win or lose, which is similar to the task objective in our project. However, our project lies on a continuous space and the strategies and actions are not as explict as that in game-based scenarios.

## Model Overview

In our project, we aim to re-implement and extend the algorithms and methodology of the paper [1]. In the model, the system receives as input a character model, a corresponding set of kinematic reference motions, and a task defined by a reward function. It then synthesizes a controller that enables the character to imitate the reference motions, while also satisfying task objectives, such as striking a target or running in a desired direction over irregular terrain. Some of the key details of the project are –

- Goal of the Policy – Reproduce desired motion in the environment using reference motion clips. The reference motion provides kinematic information in the form of target poses. The policy determines the actions to achieve the trajectory.
- States – The state $s$ describes the configuration of the character's body, with features consisting of the relative positions of each link with respect to the root (designated to be the pelvis), their rotations expressed in quaternions, and their linear and angular velocities.
- Actions - The action $a$ specifies target angles for proportional-derivative (PD) controllers that then produce the final torques applied at the joints.
- Policy – Each policy $\pi$ is represented by a neural network that maps a given state s and goal g to a distribution over action $\pi(a|s, g)$. The action distribution is modeled as a Gaussian, with a state dependent mean $\mu(s)$ specified by the network, and a fixed diagonal covariance matrix $\Sigma$ that is treated as a hyperparameter of the algorithm.
- Reward – The reward is a linear combination of the imitation objective term and the task objective term. The imitation objective incentivizes the agent to follow the reference clip. The task objective incentivizes the specific task
- Learning Algorithm - Proximal Policy optimization algorithm is used to train the networks.

## Training & Transfer Learning

In the training process, two networks are maintained, one for the policy and another for the value function. The value function is updated using target values computed with TD($\lambda$). The policy is updated using gradients computed from the surrogate objective, with advantages $\mathcal{A}_t$ computed using GAE($\lambda$). Training proceeds episodically, where at the start of each episode, an initial state $s_0$ is sampled uniformly from the reference motion, and rollouts are generated by sampling actions from the policy at every step. Each episode is simulated to a fixed time horizon or until a termination condition has been triggered.

We implemented transfer learning by freezing the initial layers of the actor and critic models. In the frame of a neural network with 2 hidden layers, weights in the first layer are kept the same as the pre-trained model while weights in the second layer is fine-tuned by training on a new dataset with motion clips of different actions.

We believe that transfer learning can be extended to allow agents to learn multiple skills easily. Transfer learning will make it easier for the model to converge to a solution.

## Experiments and Results

In our case, we used a pre-trained model of a humanoid capable of walking. The agent was fine-tuned to learn running and zombie walking. From figure 1, it can be seen that the transfer learning techniques really speed up the training progress.

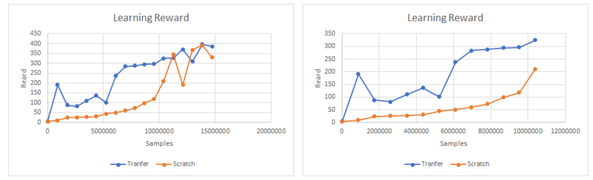


Figure 1: The training reward v.s. the number of sampled episodes. The blue curve corresponds to the transfer learning while the orange curve corresponds to the training from scratch.

Figure 2 and 3 are some sampled clips from our trained model of running and zombie walking with the transfer learning techniques.

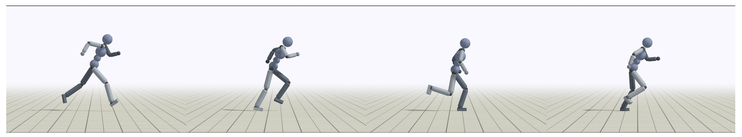

Figure 2: Sampled clips from the trained model for running

Figure 3: Sampled clips from the trained model for zombie walking

## References

[1] Peng, X. B., Abbeel, P., Levine, S., van de Panne, M.(2018)., *Deepmimic: Example-guided deep reinforcement learning of physics-based character skills*. ACM Transactions on Graphics (TOG), 37(4), 1-14.

[2] Peng, X. B., Abbeel, P., Levine, S., van de Panne, M. DeepMimic Github repository, `https://github.com/xbpeng/DeepMimic`