Model-based Reinforcement Learning and Transformer Architecture in a Humanoid Robot Environment

Brandon Woodward[1], **Kaustabh Paul**[1], **Will Kraus**[1], **and Anton Yanovich**[1]

[1]Carnegie Mellon University, Pittsburgh, USA

Abstract: In terms of humanoid robot whole-body control, modern approaches for deployment have leveraged simulated environments to train robust learned policies. Many training algorithms use several multi-layer perceptrons (MLPs) to encode learned dynamics. In contrast, much research has been done on transformers that predict tokens in a multitude of contexts using different types of training data. This paper proposes the use of transformers as a substitution for MLPs in a reinforcement learning training architecture. The performance of this addition is tested in a model-based reinforcement learning approach on a whole-body control policy for a humanoid robot and compared against the performance without the use of a transformer. We show that a transformer architecture allows a humanoid robot to reach comparable performance with less training time. In addition, we experiment with transformer pre-training and show that this significantly enhances the smoothness of robots joints transitions unrelated to the reward as well as significant improvement in training time.

Keywords: Reinforcement Learning, Whole Body Control, Humanoid Robotics

1 Introduction

Prior to the refinement of reinforcement learning techniques, whole-body control of humanoid robots primarily leveraged model predictive control and quadratic program solvers to provide constraints for contact dynamics and manipulation tasks. While these methods are stable and sample efficient, the main drawback of these controllers is their lack of generalizability and adaptability across domains [1]. Tuning these controllers has been an ongoing effort for effective hardware rollouts and has required specialized teams of engineers. However, with reinforcement learning and other techniques that leverage simulation and neural networks, policies that achieve the same level of performance can be created and tuned in simulation before deployment. This approach to whole-body control allows for real-time deployment that is robust enough for useful work applications while remaining generalizable enough to complete different tasks [2]. One of the strengths of reinforcement learning is the reliance on neural networks that capture nonlinearities in certain models, such as contact dynamics interactions and certain sim-to-real gaps. These neural networks act as function approximators and show improved capabilities over solely model-based control [3]. In the last few years, transformer-based neural network models have been used in robotics applications, such as in high-level planning and decision-making [4]. The idea of a transformer that can intuitively process complex relationships between actions and states is particularly interesting, as their applications may be useful in controlling hard-to-describe control tasks.

The present paper seeks to benchmark the performance of a model-based reinforcement learning approach with MLPs and with a selection of MLPs replaced with a transformer. The contribution leverages the generalizability of a transformer architecture, pre-trained across a number of tasks in a variety of worlds and embodiments, within an already versatile reinforcement learning algorithm. In

Course Project, 16-831 Introduction to Robot Learning, Spring 2024, CMU

the future, this approach to neural network structure interchangeability could enhance the quality of tasks completed with model-based algorithms and broaden the type of tasks that can be completed.

2 Related Work

2.1 Model-Based Reinforcement Learning

The refinement of model-based reinforcement learning methods has been a gradual process stemming from model-based control system architectures. Instead of being assigned a model as a base for control policies, model-based reinforcement learning attempts to learn a model for a system and prescribe actions upon the simulated or physical system. In particular, approaches that encode parts of the world model alongside model-specific information have been successful at generalizing across a variety of tasks due to the encoding of reward-specific information; this allows for a better intuitive understanding of the reward and tasks [5]. While this method is robust to changes in tasks and environments, the focus on unnecessary attributes of the encoded state, such as colors or shading in the simulated environment, can lead to unnecessary processing and encoding.

2.2 Transformers

Transformers are a multi-headed, encoder-decoder architecture originally used for natural language processing in large language models (LLMs). While the exact implementation of transformer structures varies across different papers, the overall goal remains the same: receive a sequence of inputs, connect inputs across the sequence using an attention structure, and abstract the inputs into the latent space repeatedly using the encoder. A decoder then predicts the outputs based on these correlated latent space representations. Transformers are particularly well-suited to robotics applications, since the combination of action, state, and reward sequences have correlations that can be adapted into positional encoding for multi-head attention via timesteps [6]. Previous work on implementing LLMs into robotics applications has focused on long-horizon planning at a high level for complex tasks that are challenging to program, such as highly structured but logistically complex cooking tasks [7]. However, this structure can be exploited for other tasks as well, as tokens of text can be interchanged for any sequence of inputs.

3 Method

The transformer architecture in this contribution draws inspiration from the decision transformer by Chen et al; the inputs to the transformer are previous action, encoded state and task, rewards, return-to-go sets, timesteps, and attention masks, while the outputs are predicted state, actions, rewards and Q-values [6]. The reinforcement learning structure to be modified by this transformer, TD-MPC2, is used for both its generalizability across different embodiments and the availability of its open-source training dataset [8]. The TD-MPC2 agent plans future actions using the MPPI algorithm, a sampling based model predictive control framework; the transformer is reconfigured to output action mean and log-variance instead in this area of TD-MPC2. Figure 1 shows how the MPPI algorithm is changed by replacing the MLPs responsible for action mean and log-variance generation with the modified decision transformer. Some of the previously mentioned inputs and outputs (e.g. timesteps, attention masks, predicted Q-values) are not expressed to simplify how the implementation is structured. The transformer is trained to not explicitly require inputs from all sources and unspecified input values are set to zero. This allows the TD-MPC2 algorithm to provide the transformer with only currently available information while still being able to function correctly. Hence, the transformer can successfully predict future action, mean, and log-variance given only the current state. For initial testing results, the action, mean, and log-variance are used to evaluate the viability of replacing an MLP with the transformer; the output architecture is kept intact to potentially replace more MLPs as a potential future work.

Pre-training of the transformer is done using the TD-MPC2 dataset MT30 [8], which has 345 million transitions across 11 different models performing 30 different tasks. The training uses a combined

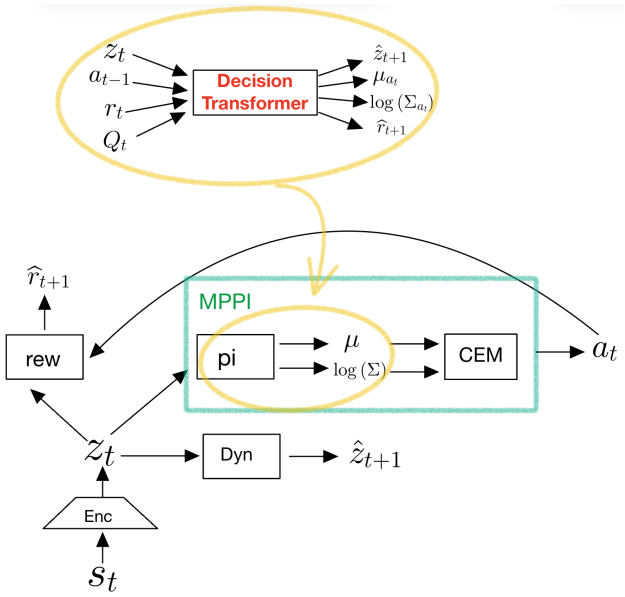

Figure 1: System architecture of transformer integration.

loss function of MSE for the encoded state and the action, soft cross entropy for the reward and Q-values, and Q-learning loss using log-likelihood and the maximum Q at a given state for the action distribution. The Q-value used in the action distribution loss is copied and detached from the original Q-value tensor to avoid any backpropagation from this portion of the loss. The pre-training also has a tuneable probability to provide the transformer with solely state state information to avoid becoming reliant on a small selection inputs like previous actions or rewards.

This hybrid structure is implemented in a whole-body control simulation from the HumanoidBench implementation, which standardized whole-body control tasks using a Unitree H1 with two Shadow Dexterous Hands in either fixed or actuated configurations [9]. A total of 27 tasks spanning locomotion, static manipulation, and dynamic manipulation tasks were organized in a variety of real-world scenarios. While there are a wide variety of tasks to decide for an intial implementation, the task *sit_simple* is used as a challenge in comparison to classical MPC-based policies, since sitting in a chair typically requires contact forces that are challenging to model.

4 Experimental Results

Initial results using a transformer initialized with random weights reach similar returns to the base TD-MPC2 algorithm with a 54% reduction in the number of training epochs. Intuitively, these results make sense because of how updates to the neural networks work. Using a normal MLP, transitions are passed into the network in batches and loss is calculated for each individual transition. However, with a transformer, the transition sequences can be linked to each other, and the action transitions are learned at a faster rate.

During pre-training, the transformer learns smooth motions that are consistent with successful reward function implementations across different embodiments, since a higher reward is consistent with smooth, fluid motions as opposed to erratic behavior. These connections are maintained through the fine-tuning from TD-MPC2. This is most noticeable in the arms of the robot, which are absent from the reward function for the *sit_simple* task. A qualitative comparison of the training results shows that the arms of the robot were relatively restrained to the torso area and had far less erratic behavior, even though there was no positive reward function for doing so. This video result is available in the Appendix via QR code.

Figure 2: Results at roughly 1 million steps in training; videos can be viewed in QR code form in the appendix.

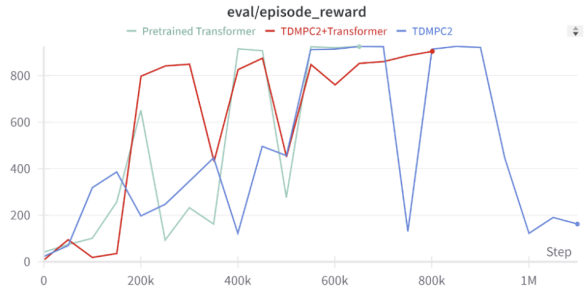

Figure 3: *episode_reward* across the pre-trained transformer, the standalone TD-MPC2 implementation, and the combined TD-MPC2 and transformer implementation. Note that the training for the algorithms was prematurely ended due to the limited computing available for the project.

The rewards reach the same level of return of around 800, but the "TD-MPC2 + Transformer" implementation reaches these returns before the pre-trained transformer. With the transformer pre-trained, the training time is reduced by 27%, while a less smooth implementation without pre-training showed a 54% reduction in training time.

5 Conclusion

The paper proposes an implementation of whole-body control for a humanoid robot using reinforcement learning with both a pre-trained and untrained transformer model. By switching to a transformer-based approach instead of using MLPs in a custom implementation of TD-MPC2, the final results show improvements over a standard reinforcement learning implementation. In total, the contribution gives credence to a possible solution to issues with reinforcement learning implementations, as multi-headed attention is suited for a sequential input of various domains for state and prior action. The transformer showed promising results when used to replace a single MLP in the TD-MPC2 algorithm, reducing the training time needed for similar results to the base implementation by a minimum of 27% with pre-training and a maximum of 54% without pre-training. The pre-trained algorithm shows more robust results at the cost of a longer online training time. Therefore, both methods shows a reduction in training time over the unmodified TD-MPC2 implementation. Future areas of study for this topic include the addition of more tasks found within the HumanoidBench paper, such as dexterous manipulation and robot athleticism. In addition, more MLPs can be replaced using the decision transformer architecture to produce smooth actions and reduced training time.

References

[1] T. Koolen, S. Bertrand, G. Thomas, T. De Boer, T. Wu, J. Smith, J. Englsberger, and J. Pratt. Design of a momentum-based control framework and application to the humanoid robot atlas. 13(1):1650007. ISSN 0219-8436, 1793-6942. doi:10.1142/S0219843616500079. URL https://www.worldscientific.com/doi/abs/10.1142/S0219843616500079.

[2] X. Cheng, Y. Ji, J. Chen, R. Yang, G. Yang, and X. Wang. Expressive whole-body control for humanoid robots. URL http://arxiv.org/abs/2402.16796.

[3] Y. Song, A. Romero, M. Mueller, V. Koltun, and D. Scaramuzza. Reaching the limit in autonomous racing: Optimal control versus reinforcement learning. 8(82):eadg1462. ISSN 2470-9476. doi:10.1126/scirobotics.adg1462. URL http://arxiv.org/abs/2310.10943.

[4] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. URL http://arxiv.org/abs/2201.07207.

[5] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap. Mastering diverse domains through world models. URL http://arxiv.org/abs/2301.04104.

[6] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch. Decision transformer: Reinforcement learning via sequence modeling. URL http://arxiv.org/abs/2106.01345.

[7] F. Joublin, A. Ceravola, P. Smirnov, F. Ocker, J. Deigmoeller, A. Belardinelli, C. Wang, S. Hasler, D. Tanneberg, and M. Gienger. CoPAL: Corrective planning of robot actions with large language models. URL http://arxiv.org/abs/2310.07263.

[8] N. Hansen, H. Su, and X. Wang. TD-MPC2: Scalable, robust world models for continuous control. URL http://arxiv.org/abs/2310.16828.

[9] C. Sferrazza, D.-M. Huang, X. Lin, Y. Lee, and P. Abbeel. HumanoidBench: Simulated humanoid benchmark for whole-body locomotion and manipulation. URL http://arxiv.org/abs/2403.10506.

A Appendix

A.1 Decision Transformer Hyperparameters

The following are our hyperparameters tuned for the selected task.

Parameter	Value
numtasks	31
latent_dim	512
action_dim	61
use_horizon_batchsize_dimensioning	False
only_state_p	0.5
multitask	True
obs	state
obs_shape	$\{'state' : [151]\}$
tasks	[0, 1, ..., 29, 'humanoid_h1hand-sit_simple-v0']
task_dim	64
num_enc_layers	2
enc_dim	2
simnorm_dim	8
encoder_dim	256
lr	1×10^{-4}
num_epochs	120
batch_size	64
clip_grad_norm	20
rho	0.5
entropy_coef	1×10^{-4}
tau	0.005
horizon	4
consistency_coef	20
reward_coef	0.1
value_coef	0.1
action_coef	0.1
log_pi_coef	0.1
num_bins	101
vmin	-10
vmax	10
bin_size	$\frac{vmax-vmin}{num\_bins-1}$

A.2 TD-MPC2 Hyperparameters

We use the same hyperparameters as the authors of the TD-MPC2 method.

Hyperparameter	Value
Planning	
Horizon (H)	3
Iterations	6 ($+2$ if $\|\mathcal{A}\| \geq 20$)
Population size	512
Policy prior samples	24
Number of elites	64
Minimum std.	0.05
Maximum std.	2
Temperature	0.5
Momentum	No
Policy prior	
Log std. min.	-10
Log std. max.	2
Replay buffer	
Capacity	$1,000,000$
Sampling	Uniform
Architecture (5M)	
Encoder dim	256
MLP dim	512
Latent state dim	512
Task embedding dim	96
Task embedding norm	1
Activation	LayerNorm + Mish
Q-function dropout rate	1%
Number of Q-functions	5
Number of reward/value bins	101
SimNorm dim (V)	8
SimNorm temperature (τ)	1
Optimization	
Update-to-data ratio	1
Batch size	256
Joint-embedding coef.	20
Reward prediction coef.	0.1
Value prediction coef.	0.1
Temporal coef. (λ)	0.5
Q-fn. momentum coef.	0.99
Policy prior entropy coef.	1×10^{-4}
Policy prior loss norm.	Moving $(5\%, 95\%)$ percentiles
Optimizer	Adam
Learning rate	3×10^{-4}
Encoder learning rate	1×10^{-4}
Gradient clip norm	20
Discount factor	Heuristic
Seed steps	Heuristic

A.3 Reward Implementation

We use the same reward function as the authors of Humanoid Bench.

$$
\begin{aligned}
\text{sitting}_x &= \text{tol}(x_{\text{robot}} - x_{\text{chair}}, (-0.19, 0.19), 0.2) \\
\text{sitting}_y &= \text{tol}(y_{\text{robot}} - y_{\text{chair}}, (0, 0), 0.1) \\
\text{sitting}_z &= \text{tol}(z_{\text{robot}}, (0.68, 0.72), 0.2) \\
\text{posture} &= \text{tol}(z_{\text{head}} - z_{\text{IMU}}, (0.35, 0.45), 0.3) \\
\text{still}_x &= \text{tol}(v_x, (0, 0), 2) \\
\text{still}_y &= \text{tol}(v_y, (0, 0), 2)
\end{aligned}
$$

$$
R(s, a) = \left(0.5 \cdot \text{sitting}_z + 0.5 \cdot \text{sitting}_x \times \text{sitting}_y\right) \times \text{upright} \times \text{posture} \times e \times \text{mean}(\text{still}_x, \text{still}_y)
$$

A.4 QR Code Video