# Accelerated Deep Reinforcement Learning of Terrain-Adaptive Locomotion Skills

**Khaled S. Refaat**
Waymo LLC
Mountain View, CA 94043
krefaat@waymo.com

**Kai Ding**
Waymo LLC
Mountain View, CA 94043
dingkai@waymo.com

## Abstract

Learning locomotion skills on dynamic terrains allows creating realistic animations without recording motion capture data. The simulated character is trained to navigate varying terrains avoiding obstacles with balance and agility. Model-free reinforcement learning has been used to develop such skills for simulated characters. In particular, a mixture of actor-critic experts (MACE) was recently shown to enable learning of such complex skills by promoting specialization and incorporating human knowledge. However, this approach still requires access to a very large number of training interactions and explorations with a computationally expensive simulator. We demonstrate how to accelerate model-free reinforcement learning to acquire terrain-adaptive locomotion skills, as well as decrease the need for large-scale exploration. We first generalize model-based value expansion (MVE) to a mixture of actor-critic experts, showing the conditions under which the method accelerates learning in this generalized setting. This motivates combining MACE with MVE resulting in the MACE-MVE algorithm. We then propose learning to predict future terrains, character states, rewards, and the probability of falling down via convolutional networks to speed-up learning using generalized MVE. We analyze our approach empirically showing that it can substantially speed-up learning of such challenging skills. Finally, we study the effect of various design choices to control for uncertainty and manage dynamics fidelity.

## 1 Introduction

Agile legged locomotion and its adaptation to varying terrains require dynamically responding to foot-contact transitions under uncertainty due to variable terrain and sensorimotor errors. In nature, animals precisely control their limbs to move on dynamic terrains while avoiding collisions and falls. In computer animation, such movement is typically created by a domain expert, or using motion capture data [24]. In recent years, model-free deep reinforcement learning combined with access to a simulator has provided a simple approach to learn novel skills on new types of terrains [21, 22, 19, 2]. In this case, we formulate the problem as a sequential decision making process, where an agent takes an action given a current state, and observes a new state and a reward as a result. The agent learns a policy, typically represented as a deep neural network, that maximizes the expected sum of discounted rewards received. This allows learning a control policy that operates directly on the high-dimensional character, action and terrain representations, without the need for hand-crafting features, or relying on domain knowledge.

Despite its appealing advantages, the sample complexity of such model-free learning of terrain-adaptive locomotion skills tends to be large, particularly due to the high dimensional states, as well as the rich function approximators that are much needed to process them; see [25]. Furthermore, high-fidelity physics simulation required to collect data can be very expensive [21]. Thus, learning

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

typically requires a very large number of interactions with a computationally intensive simulator to explore and acquire new locomotion skills. To improve the sample complexity and reduce the exploration needed, one direct approach is to design sparse character and terrain descriptors which can make learning more efficient. This, however, requires domain knowledge and provides no guarantee that the designed representations will not lose critical information.

A mixture of actor-critic experts (MACE) significantly improves learning of terrain-adaptive locomotion skills compared to other model-free reinforcement learning algorithms [21], without hand-crafting sparse representations, as it promotes specialization and makes learning easier for this class of challenging problems. Namely, MACE develops a set of individual control actors, with their associated value functions. Each actor specializes in a particular regime of the overall motion using an actor bias. After learning is done, the actor associated with the highest value function gets to execute its action. Despite its appealing advantages, the sample complexity of MACE remains to be large.

One way to improve the sample complexity is through learning a dynamics model that provides imagined data to model-free value estimation algorithms [28]. In particular, model-based value expansion (MVE) [7] is a recently proposed technique used to improve the quality of state-action value functions by providing more reliable target values to learn from. MVE was originally proposed for a single actor-critic and was shown to improve the sample complexity. However, it is not obvious if the theoretical guarantees and practical performance hold with the general case where a mixture of actor-critic experts is used, as in MACE. This calls for a generalization of MVE for the case where an actor is selected at each time step if its corresponding critic gives the maximum value.

In this paper, we generalize MVE to a mixture of actor-critic experts, showing that similar error bounds still hold in the general case. In addition, we demonstrate the conditions under which the method accelerates learning in this generalized setting. Motivated by the theoretical results, we combine MACE with MVE resulting in the MACE-MVE algorithm. We then propose a system that accelerates model-free deep reinforcement learning of terrain-adaptive locomotion skills using MACE-MVE, by learning to imagine future experiences that are utilized to speed up training. In particular, we propose to learn prediction models represented as deep convolutional networks [14] to imagine future experiences without relying on the simulator. We design the prediction models to imagine future states observed by applying actions to current states, as well as future rewards, and the probability of falling down that stops the simulation. We use the imagined states and rewards to improve the quality of the multiple state-action value functions used by the mixture of actor-critic experts via generalized MVE.

In addition, we demonstrate empirically that MACE-MVE substantially decreases the sample complexity, accelerating MACE by $19\%$ to $43\%$ on different terrains and characters, and decreases the need for large-scale exploration. This leads to acquiring better terrain-adaptive locomotion skills significantly faster than MACE. Our trained simulated characters demonstrate impressive agility and is capable of traveling for longer distances on very challenging and dynamic terrains compared to previous work. Furthermore, we conduct an ablation study for various hyper parameters and design choices. In particular, we show the importance of utilizing shorter imagination horizons that better control for uncertainty, and smaller replay buffers that create more local models. We also report that the best results are gotten when MVE is disabled after a certain number of iterations, at which dynamics fidelity becomes more crucial during later learning stages. Moreover, we experiment with utilizing the imaginary tuples for actor updates, in addition to critic updates. We show that the imaginary tuples are better used for critic updates only. Finally, in the supplementary material, we provide a video illustrating the skill acquisition progression, and the final trained policies using the proposed approach, successfully navigating challenging environments with walls, steps, gaps and slopes.

## 2 RELATED WORK

Most of the approaches developing motion from first principles have been focused on locomotion of various characters on flat terrain, e.g. [10, 26, 4, 15, 13, 6]. These methods typically rely on model-free approaches where a controller is designed and some of its parameters are optimized. We will build on previous work where the action parameters are predesigned but the controller is learned.

When the system dynamics are not known, policy gradient methods [23] and value function learning with function approximation are typically used [27]. Off-policy algorithms tend to achieve better sample complexity [18].

In [21], deep reinforcement learning was used to learn how to move a character with agility on varying terrains. Multiple tricks were used to stabilize and accelerate learning. An initial set of good actions were hard-coded to bootstrap learning by initializing the replay buffers. In addition, a mixture of actor-critic experts was utilized, each of which is encouraged to specialize in an action: running, slowing, and jumping, through an actor bias added to each actor output. We found these tricks to be effective, but the number of interactions with the simulator required to learn useful policies is still substantially large.

To accelerate learning, guided policy search has been used for bipedal walking for planar characters; e.g. [17, 16], which use trajectory optimization in a contact-invariant fashion. Model-based techniques [1, 12, 5] exhibit efficient learning but their asymptotic performance on complex tasks typically lag behind model-free methods, except with a careful control of uncertainty [3]. Model-free techniques, however, can learn challenging skills, but require a much larger number of interactions with the simualtor. In this paper, we seek to reduce the sample complexity of model-free reinforcement learning of locomotion skills on varying terrains.

Prior work has incorporated dynamics models into model-free reinforcement learning. In [9], a model is used to improve credit assignment from real trajectories. The idea of providing imagined data to model-free value estimation algorithms can be rooted back to [28]. Follow-up ideas have been proposed such as [8] where additional training data were used for a parameterized value network. In [11], imaginary rollouts were used with an inaccurate model, and a notion of uncertainty was introduced, to make use of artificial data only in cases of high uncertainty.

Model-based value expansion (MVE) [7] controls for uncertainty by only allowing imagination to a fixed depth. This improves value estimation which reduces the sample complexity needed to learn. However, MVE was originally proposed for a single actor-critic. In this paper, we generalize the theoretical analysis to a mixture of actor-critic experts to motivate equipping MACE with MVE. One key insight of our work is that for learning terrain-adaptive locomotion skills using a mixture of actor-critic experts, the bottleneck lies in learning high quality $\mathcal{Q}$-values, which we tackle by training prediction models that are used to improve the quality of the $\mathcal{Q}$-values while learning.

# 3 APPROACH

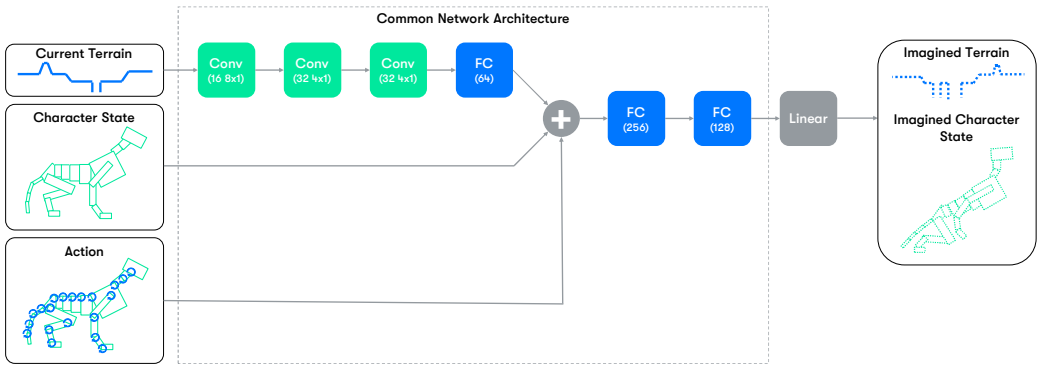
In reinforcement learning, the goal is to learn a policy $\pi(s)$ that maps a state $s \in \mathcal{S}$ to an action $a \in \mathcal{A}$. At each time step $t \in [0, \mathcal{T}]$, the agent executes an action $a_t = \pi(s_t)$ in the environment and experience a new state $s_{t+1}$ and a reward $r(s_t, a_t, s_{t+1})$. The goal is to learn a policy that maximizes the expected sum of discounted future rewards from a random initial state $S_0$: $\mathcal{V}(s_0) = r_0 + \gamma r_1 + \cdots + \gamma^{\mathcal{T}} r_{\mathcal{T}}$ where $r_i = r(s_i, a_i, s_{i+1})$ and $\gamma < 1$ to ensure the sum of discounted rewards is bounded for cases where $\mathcal{T} \to \infty$. We rewrite as: $\mathcal{V}(s_0) = r_0 + \gamma \sum_{i=1}^{\mathcal{T}} \gamma^{i-1} r_i = r_0 + \gamma \mathcal{V}(s_1)$.

The expected discounted return $\mathcal{V}$ can be optimized using different reinforcement learning algorithms. To learn locomotion skills of characters with varying terrains, deep reinforcement learning has been used to process raw high-dimensional states.

## 3.1 MACE

We first review the MACE [21] approach which we build upon in our work. MACE uses a mixture of actor-critic experts that exhibit specialization. Each actor proposes an action represented as a vector of continuous values, whereas its corresponding critic evaluates the value of executing such action from the current state, which is called the $\mathcal{Q}$-value. After the policy is learned, the actor associated with the highest $\mathcal{Q}$-value executes its action.

During training, each transition is stored as a tuple $(s_i, a_i, r_i, s_{i+1}, \mu_i)$, where $\mu_i$ indicates the index of the active actor. A tuple is stored in one of two replay buffers used for learning the actors and critics, respectively. In particular, tuples with added exploration noise are used to train the actors, whereas the remaining tuples train the critics. During a critic update, a mini batch is sampled to perform a Bellman backup, by computing targets: $y_i = r_i + \gamma \max_\mu \mathcal{Q}_\mu(s_{i+1}|\theta)$, where $\mathcal{Q}_\mu(s|\theta)$ is the $\mathcal{Q}$-value




(a)

Figure 1: The model architecture used to predict a future quantity (e.g. next state as shown in the figure, reward, probability of falling), given a state represented by the terrain $T$, the character state $C$, and the action $A$. We use *conv* and *fc* to describe a convolutional and a fully connected layer, respectively. The same architecture is repeated to predict each quantity, with the output layers and loss functions being different. For the networks predicting future states and rewards, we use the mean squared error (MSE) as the loss function. For the network predicting the probability of falling, we use a sigmoid cross-entropy loss. The usage of different networks allows each to specialize on its task.

predicted by the critic corresponding to executing the action from Actor $\mathcal{A}_\mu$. After that, the targets $y_i$ are used to update the network parameters $\theta$: $\theta \leftarrow \theta + \alpha \left(\frac{1}{n}\sum_i (y_i - \mathcal{Q}_{\mu_i}(s_i|\theta)) \frac{\partial \mathcal{Q}_{\mu_i}(s_i|\theta)}{\partial \theta}\right)$. The actors are also updated by sampling a mini batch of tuples and using a CACLA-style [29] update, by first computing: $\delta_j = y_j - \max_\mu \mathcal{Q}_\mu(s_j|\theta)$ where $y_j$ is computed using an exploratory action $a_j$. If $\delta_j > 0$, which indicates a room for improving the actor, an update is performed: $\theta \leftarrow \theta + \alpha \left(\frac{1}{n}(a_j - \mathcal{A}_{\mu_j}(s_j|\theta)) \frac{\partial \mathcal{A}_{\mu_j}(s_j|\theta)}{\partial \theta}\right)$.

### 3.2 Accelerating MACE

To improve the sample efficiency of MACE, we propose training three separate convolutional neural networks. The first one takes the current state as an input which includes the current terrain description within a certain distance from the agent and the character state, and outputs the next terrain and character state, given an action $A$. The second and the third networks take the same input and have similar architectures, but output a prediction for the reward and the probability of falling, respectively.

Figure 1 shows the model architecture when used to predict the next terrain and character state. For each neural network, we used a sequence of convolutional layers and a fully connected layer to process the current train, which outputs a terrain embedding. The character state, action, and terrain embedding are then concatenated before being processed by fully connected layers. We used the mean squared error loss to learn future states and rewards, and the sigmoid cross-entropy loss to learn the probability of falling down. Furthermore, we train the models simultaneously, with the deep RL MACE iterations, and use them to speed-up the learning of the Q-values, using a technique known as model-based value expansion [7]. In particular, when we perform the bellman backup, we do: $y_i = r_i + \sum_{t=1}^{H-1} \gamma^t \hat{r}_{i+t} + \gamma^H \max_\mu \mathcal{Q}_\mu(\hat{s}_{i+H}|\theta)$, where $\hat{r}_{i+1} \ldots \hat{r}_{i+H-1}$ and $\hat{s}_{i+H}$ are gotten by applying the prediction models via the current policy, to predict the next states and rewards. If the character is predicted to fall, we assume no future states are visited in this episode. The length of the expansion $H$ reflects the confidence in the model, and allows us to limit the noise added to targets $y_i$ due to prediction errors. In our experiments, we found that a length $H \leq 3$ to be effective for this application.

Algorithm 1 describes the MACE-MVE training procedure. Similar to [21], we use separate replay buffers to train actors and critics, which are initialized using a random policy. The prediction models are utilized to train the critics, and are being updated with each reinforcement learning iteration.

**Algorithm 1** MACE-MVE

$\theta \leftarrow$ random weights
Initialize $D_c$ and $D_a$ with tuples from a random policy.
**while** not done **do**
  **for** step = $1, \ldots, m$ **do**
    Using the simulator, add a tuple to one of the replay buffers as in MACE.
  Update Critics:
  Sample minibatch of tuples $\tau_i = (s_i, a_i, r_i, s_{i+1}, \mu_i)$ from $D_c$
  Apply the prediction models $k$ times
  $y_i \leftarrow r_i + \sum_{t=1}^{H-1} \gamma^t \hat{r}_{i+t} + \gamma^H \max_\mu \mathcal{Q}_\mu(\hat{s}_{i+H}|\theta)$
  $\theta \leftarrow \theta + \alpha \left( \frac{1}{n} \sum_i (y_i - \mathcal{Q}_{\mu_i}(s_i|\theta)) \frac{\partial \mathcal{Q}_{\mu_i}(s_i|\theta)}{\partial \theta} \right)$
  Update Actors:
  Sample minibatch of tuples $\tau_j = (s_j, a_j, r_j, s_{j+1}, \mu_j)$ from $D_a$
  **for** each $\tau_j$ **do**
    $y_j \leftarrow r_j + \gamma \max_\mu \mathcal{Q}_\mu(s_{j+1}|\theta)$ $\delta_j \leftarrow y_j - \max_\mu \mathcal{Q}_\mu(s_j|\theta)$
    **if** $\delta_j > 0$ **then**
      $\theta \leftarrow \theta + \alpha \left( \frac{1}{n} (a_j - \mathcal{A}_{\mu_j}(s_j|\theta)) \frac{\partial \mathcal{A}_{\mu_j}(s_j|\theta)}{\partial \theta} \right)$
  Update Predictors:
  **for** step = $1, \ldots, s$ **do**
    Sample a mini batch of tuples from $D_a$ and $D_c$
    Train the prediction networks via the batches.

### 3.3 MACE-MVE Analysis

We generalize the analysis in [7] to the case where the $\mathcal{Q}$-values are gotten as a maximization of an ensemble of critics. In particular, we describe the conditions under which the MACE-MVE estimates are better than the vanilla MACE version, with respect to the mean squared error, defined as: $\text{MSE}_\nu(\max_\mu \mathcal{Q}_\mu) = \mathbb{E}_{S \sim \nu} \left[ \left( \max_\mu \mathcal{Q}_\mu(S) - \max_\mu {\mathcal{Q}_\mu}^\pi(S) \right)^2 \right]$, where $\nu$ is the distribution of states in the environment, and $\max_\mu {\mathcal{Q}_\mu}^\pi(S)$ is the maximization of the ensemble of critics under the current policy $\pi$.

To simplify the notation, let: $\max_\mu \mathcal{Q}_\mu(S) = \mathcal{T}(S)$. We first assume that the dynamics, reward, and falling models are perfect. Consider $\hat{\mathcal{T}}_H(s_0)$ to be the estimate while using MVE for a horizon $H$, we then have: $\hat{\mathcal{T}}_H(s_0) - \mathcal{T}^\pi(s_0) = \gamma^H \left( \hat{\mathcal{T}}(s_H) - \mathcal{T}^\pi(s_H) \right)$. As the dynamics model is assumed to be perfect, the rewards up to $H$ cancel out, resulting in an MSE for MACE-MVE equivalent to: $\text{MSE}_\nu(\hat{\mathcal{T}}_H) = \gamma^{2H} \, \text{MSE}_{(f^\pi)^H \nu}(\hat{\mathcal{T}})$, where $(f^\pi)^H \nu$ denotes the state visitation distribution after the forward dynamics model is used $H$ times starting from random states.

We now show that a similar condition holds in the case where the prediction models are not perfect. In particular, we generalize the model-based value expansion error theorem [7] to the ensemble of critics case to apply it to MACE.

**Theorem 3.1 (MACE Model-Based Expansion Error)** *Let $s_t, a_t, r_t$ be the states, actions, and rewards gotten by rolling out $\pi$ under the true system, whereas $\hat{s}_t, \hat{a}_t, \hat{r}_t$ be those gotten from the learned prediction models. The reward function $r$ be $L_r$-Lipschitz and the value function $\mathcal{Q}^\pi$ be $L_\mathcal{Q}$-Lipschitz. If the learned dynamics model error is bounded: $\max_{t \in [H]} \mathbb{E} \left[ \|\hat{s}_t - s_t\|^2 \right] \leq \epsilon^2$, and the dynamics model error is less than that of the ensemble of critics up to a constant $\alpha$: $\epsilon < \alpha \sqrt{\mathbb{E}(\hat{\mathcal{T}}(\hat{s}_H) - \mathcal{T}^\pi(\hat{s}_H))^2}$. We then have: $\text{MSE}_\nu(\hat{\mathcal{T}}_H) \leq k_1^2 \epsilon^2 + \gamma^H k_2 \, \text{MSE}_{(\hat{f}^\pi)^H \nu}(\hat{\mathcal{T}})$ where $k_1^2 = c_1^2 + 2\gamma^H c_1 L_\mathcal{Q} + \gamma^{2H} {L_\mathcal{Q}}^2$ and $k_2 = 2\alpha c_1 \sqrt{(1+c)} + \gamma^H (1+c)$.*

**Proof:** We generalize the model-based value expansion proof [7] to the case where an ensemble of critics is used. We first have: $\hat{\mathcal{T}}_H(s_0) = \sum_{t=0}^{H-1} \gamma^t \hat{r}_t + \gamma^H \hat{T}(\hat{s}_H)$

Thus, $(\hat{\mathcal{T}}_H(s_0) - \mathcal{T}^\pi(s_0))^2$ can be rewritten as: $\left(\left(\hat{M} - M\right) - \gamma^H \left(\hat{\mathcal{T}}(\hat{s}_H) - \mathcal{T}^\pi(s_H)\right)\right)^2$ where $\hat{M} = \sum_{t=0}^{H-1} \gamma^t \hat{r}_t$ and $M = \sum_{t=0}^{H-1} \gamma^t r_t$. Using Cauchy-Schwarz inequality, we have:

$$\begin{aligned}\underset{\nu}{\text{MSE}}(\hat{\mathcal{T}}_H) \leq \mathbb{E}(\hat{M} - M)^2 \\ + 2\gamma^H \sqrt{\mathbb{E}(\hat{M} - M)^2 \, \mathbb{E}(\hat{\mathcal{T}}(\hat{s}_H) - \mathcal{T}^\pi(s_H))^2} \\ + \gamma^{2H} \, \mathbb{E}(\hat{\mathcal{T}}(\hat{s}_H) - \mathcal{T}^\pi(s_H))^2\end{aligned}$$

Furthermore, using Hölder's inequality, where $i, j$ are each over $[H] - 1$: $\mathbb{E}(\hat{M} - M)^2 \leq \sum_{i,j} \gamma^{(i+j)} \sqrt{\mathbb{E}\left(\hat{r}_i - r_i\right)^2 \mathbb{E}\left(\hat{r}_j - r_j\right)^2}$. In addition, assuming: $\|\hat{r}_i - r_i\| \leq L_r \|\hat{s}_i - s_i\|$, we then have: $\mathbb{E}\left(\hat{r}_i - r_i\right)^2 \leq L_r^2 \, \mathbb{E}\|\hat{s}_i - s_i\|^2$ which gives: $\mathbb{E}\left(\hat{r}_i - r_i\right)^2 \leq L_r^2 \epsilon^2$. Thus, we have $\mathbb{E}(\hat{M} - M)^2 \leq c_1^2 \epsilon^2$ where $c_1$ is linear in $L_r$.

Note that: $\sum_{i,j} \gamma^{(i+j)} = (1 - \gamma)^{-2}$ for $\gamma < 1$. Furthermore: $\hat{\mathcal{T}}(\hat{s}_H) - \mathcal{T}^\pi(s_H) = (\hat{\mathcal{T}}(\hat{s}_H) - \mathcal{T}^\pi(\hat{s}_H)) - (\mathcal{T}^\pi(s_H) - \mathcal{T}^\pi(\hat{s}_H))$. Similar to [7], using Cauchy-Schwarz:

$$\begin{aligned}&\mathbb{E}(\hat{\mathcal{T}}(\hat{s}_H) - \mathcal{T}^\pi(s_H))^2 \\ &\quad \leq \mathbb{E}(\hat{\mathcal{T}}(\hat{s}_H) - \mathcal{T}^\pi(\hat{s}_H))^2 \\ &\qquad + 2\sqrt{\mathbb{E}(\hat{\mathcal{T}}(\hat{s}_H) - \mathcal{T}^\pi(\hat{s}_H))^2 \, \mathbb{E}(\mathcal{T}^\pi(\hat{s}_H) - \mathcal{T}^\pi(s_H))^2} \\ &\qquad + \mathbb{E}(\mathcal{T}^\pi(\hat{s}_H) - \mathcal{T}^\pi(s_H))^2.\end{aligned}$$

In addition, we have: $\mathbb{E}(\hat{\mathcal{T}}(\hat{s}_H) - \mathcal{T}^\pi(\hat{s}_H))^2 = \text{MSE}_{(\hat{f}^\pi)^H \nu}(\hat{\mathcal{T}})$. Recall that: $\mathcal{T}(S) = \max_\mu \mathcal{Q}_\mu(S)$. We use the property that the maximum of Lipschitz functions is Lipschitz, to generalize to the maximum of critics case: $\mathbb{E}(\mathcal{T}^\pi(\hat{s}_H) - \mathcal{T}^\pi(s_H))^2 \leq L_{\mathcal{Q}}^2 \epsilon^2$

We then have: $\mathbb{E}(\hat{\mathcal{T}}(\hat{s}_H) - \mathcal{T}^\pi(s_H))^2 \leq \text{MSE}_{(\hat{f}^\pi)^H \nu}(\hat{\mathcal{T}}) + 2L_{\mathcal{Q}}\epsilon\sqrt{\text{MSE}_{(\hat{f}^\pi)^H \nu}(\hat{\mathcal{T}})} + L_{\mathcal{Q}}^2 \epsilon^2$

and since: $\epsilon < \alpha\sqrt{\mathbb{E}(\hat{\mathcal{T}}(\hat{s}_H) - \mathcal{T}^\pi(\hat{s}_H))^2}$, we have:

$\mathbb{E}(\hat{\mathcal{T}}(\hat{s}_H) - \mathcal{T}^\pi(s_H))^2 \quad \leq \text{MSE}_{(\hat{f}^\pi)^H \nu}(\hat{\mathcal{T}}) + 2L_{\mathcal{Q}}\alpha\text{MSE}_{(\hat{f}^\pi)^H \nu}(\hat{\mathcal{T}}) + L_{\mathcal{Q}}^2 \epsilon^2.$

$\mathbb{E}(\hat{\mathcal{T}}(\hat{s}_H) - \mathcal{T}^\pi(s_H))^2 \leq (1 + c)\,\text{MSE}_{(\hat{f}^\pi)^H \nu}(\hat{\mathcal{T}}) + L_{\mathcal{Q}}^2 \epsilon^2$ where $c = 2L_{\mathcal{Q}}\alpha$.

In addition:

$2\gamma^H \sqrt{\mathbb{E}(\hat{M} - M)^2 \, \mathbb{E}(\hat{\mathcal{T}}(\hat{s}_H) - \mathcal{T}^\pi(s_H))^2} \leq 2\gamma^H c_1 \epsilon \sqrt{\mathbb{E}(\hat{\mathcal{T}}(\hat{s}_H) - \mathcal{T}^\pi(s_H))^2}$

$2\gamma^H \sqrt{\mathbb{E}(\hat{M} - M)^2 \, \mathbb{E}(\hat{\mathcal{T}}(\hat{s}_H) - \mathcal{T}^\pi(s_H))^2} \leq 2\gamma^H c_1 \epsilon \left(\sqrt{(1 + c)\,\text{MSE}_{(\hat{f}^\pi)^H \nu}(\hat{\mathcal{T}})} + \sqrt{L_{\mathcal{Q}}^2 \epsilon^2}\right)$

$2\gamma^H \sqrt{\mathbb{E}(\hat{M} - M)^2 \, \mathbb{E}(\hat{\mathcal{T}}(\hat{s}_H) - \mathcal{T}^\pi(s_H))^2} \leq 2\gamma^H \left(\alpha c_1 \sqrt{(1 + c)}\,\text{MSE}_{(\hat{f}^\pi)^H \nu}(\hat{\mathcal{T}}) + c_1 L_{\mathcal{Q}} \epsilon^2\right)$

$$\begin{aligned}\underset{\nu}{\text{MSE}}(\hat{\mathcal{T}}_H) \leq c_1^2 \epsilon^2 + 2\gamma^H \left(\alpha c_1 \sqrt{(1 + c)} \underset{(\hat{f}^\pi)^H \nu}{\text{MSE}}(\hat{\mathcal{T}}) + c_1 L_{\mathcal{Q}} \epsilon^2\right) \\ + \gamma^{2H} \left((1 + c) \underset{(\hat{f}^\pi)^H \nu}{\text{MSE}}(\hat{\mathcal{T}}) + L_{\mathcal{Q}}^2 \epsilon^2\right)\end{aligned}$$

$\text{MSE}_\nu(\hat{\mathcal{T}}_H) \leq k_1^2 \epsilon^2 + \gamma^H k_2 \, \text{MSE}_{(\hat{f}^\pi)^H \nu}(\hat{\mathcal{T}})$

where $k_1^2 = c_1^2 + 2\gamma^H c_1 L_{\mathcal{Q}} + \gamma^{2H} {L_{\mathcal{Q}}}^2$ and $k_2 = 2\alpha c_1 \sqrt{(1 + c)} + \gamma^H (1 + c)$

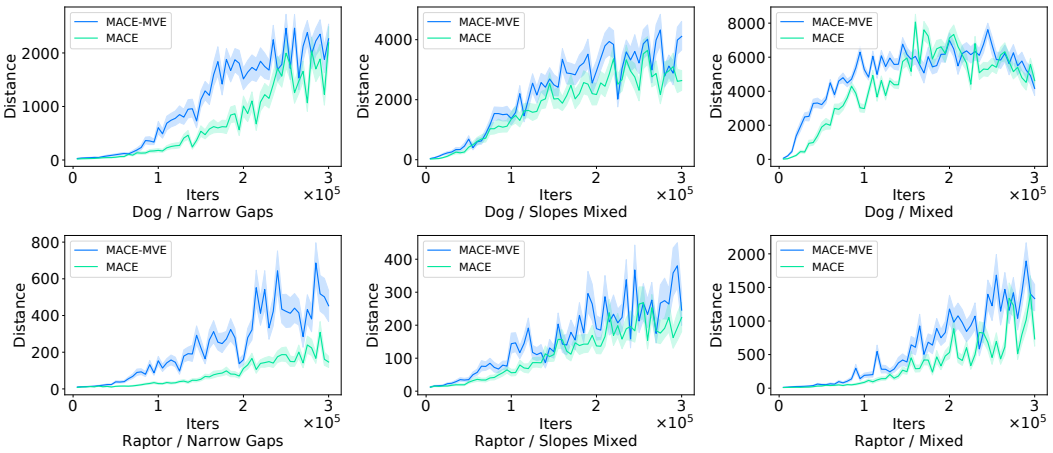



Figure 2: A comparison showing the performance of different approaches with the number of learning iterations, which is measured as distance traveled by the character using the learned policies over 100 runs. Two characters are used: Dog and Raptor, and environments are randomly generated from three environment types: Narrow Gaps, Slopes Mixed, and Mixed.

## 4 EXPERIMENTAL RESULTS

In this section, we test the benefit of the proposed approach (MACE-MVE) in accelerating the acquisition of terrain-adaptive locomotion skills. In particular, we test the following hypotheses: 1) MACE-MVE results in learned policies that can travel for longer distances than MACE, when run for the same number of iterations. 2) MACE-MVE would require less exploration than MACE, as the prediction models can help imagine unexplored states. Less exploration can take the form of decreasing the need for bootstrapping large replay buffers with tuples from a random policy, as well as decreasing the need for exploration while learning. Furthermore, we study the effect of different hyper parameters on MACE-MVE, such as: 1) imagination horizon, 2) replay buffer size, 3) activating MVE for a variable number of iterations, and 4) using imaginary tuples for actor updates; and show some qualitative results.

**Experimental Setup**: as in [21], we use 3 actor-critic experts for all policies. Separate policies are learned for each combination of character and terrain type. To evaluate the performance of a policy, we measure the distance traveled by the character averaged over 100 random runs. Given the terrain type, the actual terrain in each run is randomly generated based on predefined parameters, same as in [21]. We use the MACE implementation, which is available at [20], and modify it to implement the proposed approach. We compare MACE [21] against MACE-MVE, which combines the MACE approach with model-based value expansion to speed-up the learning of the $\mathcal{Q}$-values. Namely, MACE-MVE can be viewed as an accelerated version of MACE, which relies on the prediction networks shown in Figure 1 to speed-up learning. The prediction models are trained simultaneously with the MACE reinforcement learning iterations.

We simulate two characters: Dog and Raptor, and generate random environments from three types of terrains: Narrow Gaps, Mixed and Slopes Mixed. The Narrow Gaps terrain contains frequent gaps. The Mixed terrain contains walls, steps, and gaps, whereas the Slopes Mixed terrain contains slopes in addition; see Figure 4 for examples of these terrains. A character can see 10 meters ahead, and, therefore, gets to perceive an environment gradually as they traverse which requires agility to surprises, such as walls or gaps. We emphasize that while the steps, walls and gaps are randomly placed, there is a value from learning to predict future terrains, as a small portion of the obstacle in sight could be used to infer the entire obstacle. To be able to learn such skills, reinforcement learning algorithms typically require a very large number of interactions with the environment. We run both approaches for 300k iterations on an AMD Ryzen 9 3900X (24 threads on 12 cores). For MVE, we used $H = 3$, batch size of 32, and a dynamics replay buffer of 10k.

**Accelerating MACE**: we found that MACE-MVE decreases the sample complexity required to achieve a certain character performance, characterized by traveled distance, accelerating MACE by $19\%$ to $43\%$ on different terrains and characters. As a result, after a certain number of iterations,

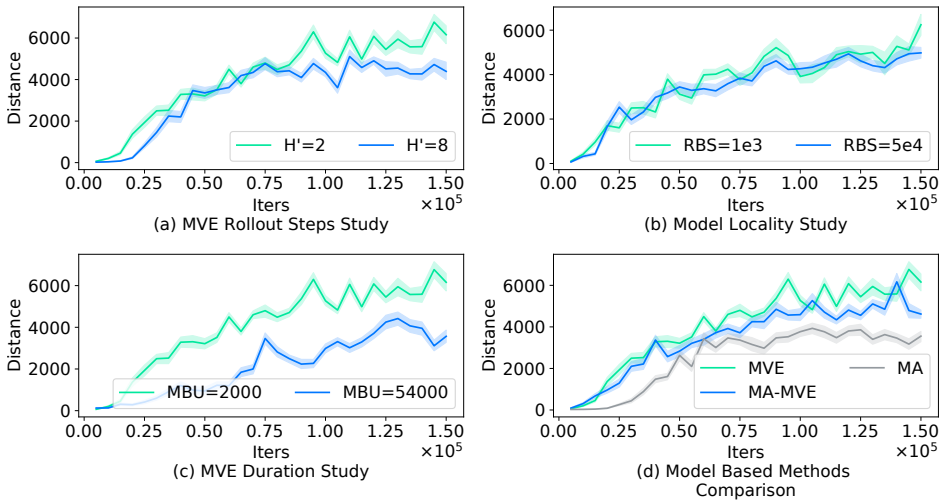



Figure 3: The average distance traveled by the dog on a mixed terrain while varying different design choices: a) various rollout steps (H). b) different replay buffer sizes used to learn the dynamics model (RBS). c) various number of iterations where MVE is enabled, i.e. iterations where model-based updates take place (MBU). d) different methods for utilizing the dynamics models. In all studies, each point is an average over 100 runs.

the character trained with MACE-MVE typically travels for longer distances than that trained with MACE, as shown in Figure 2. This agrees with the hypothesis that the prediction models help speed up learning by improving the quality of the $\mathcal{Q}$-values. In addition, for MACE-MVE, we initialized the actor and critic replay buffers with just 1k tuples, as opposed to 50k which was required by MACE to bootstrap training. Furthermore, in MACE, the parameters used to control exploration had to be linearly annealed over 50k iterations to enable learning. In MACE-MVE, we managed to do the same annealing at a much faster rate over 2k iterations. We otherwise used the same MACE hyper parameters of [21]. This shows that MACE-MVE requires much less exploration than MACE, and still gets better policies. Finally, we disable the MVE updates after 2k iterations, which we discuss in the ablation study.

**Execution Time**: The execution time of an iteration is dominated by the expensive simulator, and therefore learning in fewer iterations automatically translates into a similar reduction in wall time. Moreover, the prediction neural networks described in Figure 1 are essentially modest in size, and run very efficiently on a GPU. We therefore report the distance traveled by the character using the learned policies with respect to iterations, rather than time. The wall time for MACE was in the order of days which is problematic and expensive, given the large number of hyper parameters to experiment with.

**Imagination Horizon**: To study the effect of the imagination horizon used, we experimented with MACE-MVE to train the dog on the mixed terrain while varying the imagination horizon used. We found that shorter horizons $H' \leq 4$ lead to faster learning, where $H'$ is the imagination horizon used by MVE: $H' = H - 1$. On the other hand, a larger $H' \geq 8$ degrades performance after learning for a large number of iterations, when model fidelity becomes crucial. This shows the necessity to control for the uncertainty through shorter horizons when MVE is used [7]. In Figure 3a, we show distance traveled with $H'$ set to 2 and 8 only for clarity. $H' = 1$ and 4 showed comparable performance to 2.

**Replay Buffer Size**: Furthermore, we varied the size of the replay buffer used to learn the dynamics model. A smaller replay buffer emphasizes the importance of more recent tuples, and therefore creates a more local model. We found that a large replay buffer of size $5 \times 10^4$ examples can degrade performance. This agrees with insights from [8], where local refitted models proved effective in accelerating learning. Figure 3b illustrates distance traveled while using a replay buffer of size $5 \times 10^4$ and $10^3$. We note that buffer sizes of $5 \times 10^3$ and $10^4$ showed comparable performance to $10^3$ and are omitted for clarity.

**MVE Lifetime**: We experimented with running MVE for a variable number of iterations, and turning it off after that. We found that enabling MVE for $\geq 54 \times 10^3$ iterations degrades performance. We

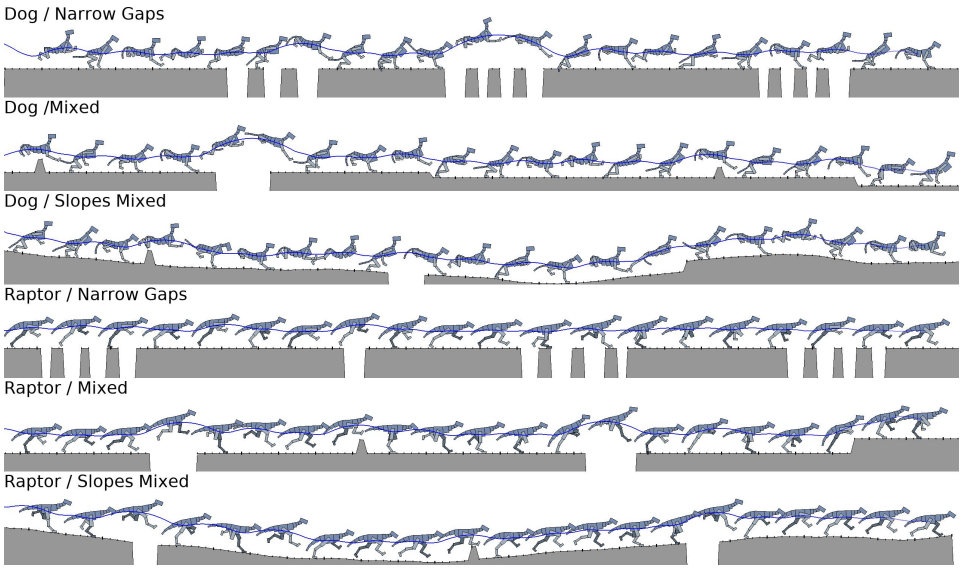



Figure 4: Qualitative examples of the simulated characters successfully traversing different terrains with policies trained using MACE-MVE.

hypothesize that the reason could be that as the $\mathcal{Q}$-function becomes more accurate, dynamics fidelity becomes more crucial during learning, which agrees with insights in [8]. Figure 3c shows distance traveled with MVE enabled for 2k and 54k iterations. Enabling MVE for 6k and 18k iterations showed comparable performance to 2k, and are omitted for clarity. Interestingly, the speed-up effect of MVE appears to last beyond its lifetime, as shown in Figure 2 (i.e. after 2k iterations with MVE enabled). Namely, MACE-MVE continues to learn faster than MACE even after MVE is disabled. This underscores the importance of early iterations in the learning procedure.

**Imaginary Tuples Usage**: Finally, we used the dynamics models to train the actors on imaginary tuples. We call this approach MA for model-based actor update. In brief, we applied random perturbations to actions and utilized the prediction models to create imaginary tuples that are added to the actor replay buffer. We combined MA with MVE resulting in MA-MVE. In all approaches, we used MACE and equipped it with each of MVE, MA and MA-MVE. We found that MVE by itself achieves the best results as shown in Figure 3d. This shows that the imagined tuples are best used to improve the critics rather than the actors.

**Qualitative Results**: In Figure 4, we show qualitative examples of the simulated characters traversing challenging terrains, while being controlled with the MACE-MVE policy. We provide a video in the supplementary material showing the skill acquisition progression, and the final trained policies.

## 5 CONCLUSIONS

We demonstrated how to accelerate deep RL for acquiring locomotion skills on highly dynamic and challenging terrains. Our approach relies on learning to predict the future and utilizing imaginary tuples. To achieve this, we first generalized model-based value expansion (MVE) analysis to an ensemble of critics, demonstrating that the MVE error bounds still hold in the general case. Motivated by the theoretical results, we equipped MACE with generalized MVE and investigated how to accelerate learning of locomotion skills on dynamic terrains. Our system utilizes efficient convolutional networks and can accelerate learning by $19\%$ to $43\%$ resulting in policies that can travel for longer distances compared to previous work. Furthermore, we showed that our approach requires much less exploration. The computational time of an iteration is dominated by the expensive physics simulator, and therefore decreasing the sample complexity leads to a corresponding decrease in training time. In addition, we demonstrated the necessity for using shorter imagination horizons and smaller replay buffers, as well as turning off MVE in later learning stages when dynamics fidelity becomes crucial. Finally, we reported qualitative results showing the impressive agility of the learned policies.

## References


[1] Christopher G. Atkeson and Juan Carlos Santamaria. A comparison of direct and model-based reinforcement learning. In *International Conference on Robotics and Automation*, pages 3557–3564. IEEE Press, 1997.

[2] Glen Berseth, Cheng Xie, Paul Cernek, and Michiel van de Panne. Progressive reinforcement learning with distillation for multi-skilled motion control. In *Proc. International Conference on Learning Representations*, 2018.

[3] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 4754–4765. Curran Associates, Inc., 2018.

[4] Stelian Coros, Philippe Beaudoin, and Michiel Panne. Generalized biped walking control. *ACM Trans. Graph.*, 29, 07 2010.

[5] Marc Deisenroth and Dieter Fox. Gaussian processes for data-efficient learning in robotics and control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37:408–423, 02 2015.

[6] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *ICML*, 2016.

[7] Vladimir Feinberg, Alvin Wan, Ion Stoica, Michael I. Jordan, Joseph Gonzalez, and Sergey Levine. Model-based value estimation for efficient model-free reinforcement learning. *ArXiv*, abs/1803.00101, 2018.

[8] Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. Continuous deep q-learning with model-based acceleration. 03 2016.

[9] Nicolas Heess, Greg Wayne, David Silver, Timothy Lillicrap, Yuval Tassa, and Tom Erez. Learning continuous control policies by stochastic value gradients. 10 2015.

[10] Jessica K. Hodgins, Wayne L. Wooten, David C. Brogan, and James F. O'Brien. Animating human athletics. In *SIGGRAPH '95*, 1995.

[11] Gabriel Kalweit and Joschka Boedecker. Uncertainty-driven imagination for continuous deep reinforcement learning. In *CoRL*, 2017.

[12] J. Kocijan, R. Murray-Smith, C. E. Rasmussen, and A. Girard. Gaussian process model based predictive control. In *Proceedings of the 2004 American Control Conference*, volume 3, pages 2214–2219 vol.3, 2004.

[13] N. Kohl and Peter Stone. Policy gradient reinforcement learning for fast quadrupedal locomotion. volume 3, pages 2619 – 2624 Vol.3, 01 2004.

[14] Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. Object recognition with gradient-based learning. In *Shape, Contour and Grouping in Computer Vision*, 1999.

[15] Yoonsang Lee, Sungeun Kim, and J. Lee. Data-driven biped control. *ACM Trans. Graph.*, 29, 07 2010.

[16] S. Levine and P. Abbeel. Learning neural network policies with guided policy search under unknown dynamics. 2:1071–1079, 01 2014.

[17] Sergey Levine and Vladlen Koltun. Learning complex neural network policies with trajectory optimization. 06 2014.

[18] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In Yoshua Bengio and Yann LeCun, editors, *ICLR*, 2016.

[19] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics (Proc. SIGGRAPH 2018)*, 37(4), 2018.

[20] Xue Bin Peng, Glen Berseth, and Michiel van de Panne. Implementation provided for the paper: Terrain-adaptive locomotion skills using deep reinforcement learning, 2016.

[21] Xue Bin Peng, Glen Berseth, and Michiel van de Panne. Terrain-adaptive locomotion skills using deep reinforcement learning. *ACM Transactions on Graphics (Proc. SIGGRAPH 2016)*, 35(4), 2016.

[22] Xue Bin Peng and Michiel van de Panne. Learning locomotion skills using deeprl: Does the choice of action space matter? In *Proc. ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 2017.

[23] Jan Peters and Stefan Schaal. Policy gradient methods for robotics. volume 2006, pages 2219 – 2225, 11 2006.

[24] Katherine Pullen and Christoph Bregler. Motion capture assisted animation: Texturing and synthesis. *ACM Transactions on Graphics*, 21, 07 2002.

[25] John Schulman, Sergey Levine, Philipp Moritz, Michael Jordan, and Pieter Abbeel. Trust region policy optimization. 02 2015.

[26] Kwang Sok, Manmyung Kim, and J. Lee. Simulating biped behaviors from human motion data. *ACM Trans. Graph.*, 26:107, 07 2007.

[27] Richard Sutton, David Mcallester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Adv. Neural Inf. Process. Syst*, 12, 02 2000.

[28] Richard S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *In Proceedings of the Seventh International Conference on Machine Learning*, pages 216–224. Morgan Kaufmann, 1990.

[29] H. van Hasselt and M. A. Wiering. Reinforcement learning in continuous action spaces. In *2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning, Honolulu, HI, 2007, pp. 272-279.*, 2007.