Journal of Machine Learning Research 24 (2023) 1-26 Submitted 10/21; Revised 1/23; Published 3/23

Towards Learning to Imitate from a Single Video Demonstration

Glen Berseth *Université de Montréal, Mila Quebec AI Institute, Institut Courtois, and Canada CIFAR AI Chair* GLEN.BERSETH@MILA.QUEBEC

Florian Golemo *Mila Quebec AI Institute* FGOLEMO@GMAIL.COM

Christopher Pal *Polytechnique Montréal, Mila Quebec AI Institute, ServiceNow Research, and Canada CIFAR AI Chair* CHRISTOPHER.PAL@SERVICENOW.COM

Editor: George Konidaris

Abstract

Agents that can learn to imitate behaviours observed in video – *without having direct access to internal state or action information of the observed agent* – are more suitable for learning in the natural world. However, formulating a reinforcement learning (RL) agent that facilitates this goal remains a significant challenge. We approach this challenge using contrastive training to learn a reward function by comparing an agent's behaviour with a single demonstration. We use a Siamese recurrent neural network architecture to learn rewards in space and time between motion clips while training an RL policy to minimize this distance. Through experimentation, we also find that the inclusion of multi-task data and additional image encoding losses improve the temporal consistency of the learned rewards and, as a result, significantly improve policy learning. We demonstrate our approach on simulated humanoid, dog, and raptor agents in 2D and quadruped and humanoid agents in 3D. We show that our method outperforms current state-of-the-art techniques and can learn to imitate behaviours from a single video demonstration.

Keywords: Reinforcement Learning, Deep Learning, Imitation Learning

1. Introduction

Imitation learning gives an agent the ability to reproduce the behaviours and skills of other agents through demonstrations (Blakemore and Decety, 2001). These demonstrations act as a type of explicit communication, informing the agent of the desired behaviour. However, real-world agents such as robots do not generally have access to the type of information needed by many imitation learning methods, such as internal state information or the executed actions of a demonstration. We also want a solution that can learn to imitate even if an observed agent has a different appearance or dynamics in the demonstration compared to the agent that is tasked with learning from the demonstration. In the same way that human children

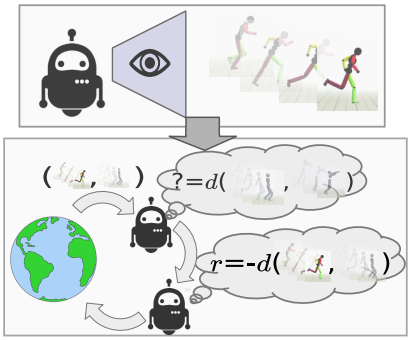


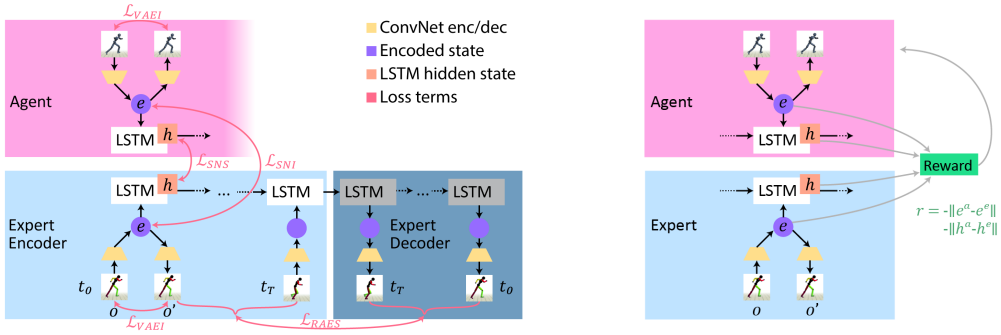
Figure 1: The agent visually observes behaviour and then collects experience to train a distance function $d()$, used to learn how to reproduce that behaviour with RL.

can learn to imitate adults by observing them, we want more versatile agents who can learn to imitate desired behaviours, given only a few examples. This type of visual-imitation is depicted in Figure 1 with an agent that learns and minimizes the distance between visual demonstrations. If we can construct agents that can learn to imitate from this kind of easy-to-obtain but noisy data, they would be much more flexible for learning via imitation in the real world.

While imitating a behaviour from observation is natural to many agents in the real world, it poses many learning challenges. Behaviour cloning (BC) methods often require expert action data making them impossible to use in more natural problem settings where only image observations are available. Also, in the real world, the demonstration agent often has different dynamics, meaning that an exact copy of the demonstration is impossible, and the learning agent must do its best to replicate the observed behaviour under *its own* dynamics. How can we train an agent to reliably imitate another agent with potentially different dynamics, given only image data from the demonstration agent? Learning a well-formed and smooth distance between observed behaviours can be used as the reward for a reinforcement learning agent. However, given the partially observed nature of the demonstration data, learning a reasonable distance function is challenging. To compensate for limited and noisy data, a method that allows us to incorporate additional offline data, potentially from other behaviours/tasks, will increase data efficiency while training a model that understands the comparative landscape of a larger behaviour space.

To realize the data-efficient and task-independent method described above, we train a recurrent Siamese comparator to capture the partial information from each image and use this comparator to compute distances used for rewards for an RL agent. We train this comparator model with off-policy data to learn distances between sequences of images (videos). This offline training makes it possible to pretrain and include data from additional behaviours/tasks to increase model robustness. Auto-encoding losses are added, shown in Fig. 2a, at different levels of granularity to increase the smoothness of the learned distance landscape. Our model learns two latent distance predictors in parallel. These two latent distance predictors are shown in Fig. 2b and allow us to compute distances between individual images and between sequences. The *image-to-image* latent distance rewards the agent for precisely matching the example behaviour. In contrast, the learned *sequence-to-sequence* latent representation provides additional reward when the agent is not just matching the desired behaviour exactly but also when the agent is replicating portions of the observed demonstration, for example, if the agent currently has different timing than the demonstration. Our results show that the robustness and smoothness gained from the combination of losses improve training efficiency and final policy quality.

Our contribution consists of a new visual-imitation learning method for RL based on visual comparisons and the specific architectures and training procedures discussed in more detail throughout the paper. We showcase that our approach enables agents to learn a large variety of challenging behaviours that include walking, running and jumping. We perform experiments for multiple simulated robots in both 2D and 3D, including simulations for training quadruped robots and a humanoid with 38 degrees of freedom (DoF). For many of these imitation tasks, our method "visual-imitation with reinforcement learning" (VIRL) is able to imitate the given skill using only a single observed demonstration.



(a) Losses for training the encoders/decoders.

(b) Reward generation for the agent.

Figure 2: VIRL learns a distance function (2a) and then uses that distance function as a reward function for RL (2b). At the current timestep, observations (**o**) of the reference motion and the agent are encoded (**e**) and fed into long short-term memory (LSTM)s (leading to hidden states **h**). Fig. 2a shows how the reward model is trained using both Siamese and autoencoder (AE) losses. There are variational autoencoder (VAE) reconstruction losses on static images ($\mathcal{L}_{VAEI}$), sequence-to-sequence AE losses ($\mathcal{L}_{RAES}$), one for the reference and one for the agent (which we do not show in pink to simplify the figure). There is a Siamese loss between encoded images ($\mathcal{L}_{SNI}$) and a Siamese loss that is computed between encoded states over time ($\mathcal{L}_{SNS}$). Fig. 2b shows how the reward is calculated **at every timestep** using the combination of the learned encodings.

2. Related Work

We group the most relevant prior work based on the type and quantity of data needed to perform imitation learning. The first group consists of GAIL (Ho and Ermon, 2016) and related methods, which require access to expert policies, states, actions, and require large quantities of expert data. In the second group, the need for expert action data is relaxed in methods like generative adversarial imitation from observation (GAIfO) (Torabi et al., 2018b), but still, requires an expert policy to repeatedly generate data. The third group avoids the need for ground truth states favouring images that are easier to obtain (Brown et al., 2019, 2020). These methods still require many examples of data from a policy trained on an agent in the same simulation with the same dynamics. Lastly, in a fourth group, the need for multiple demonstrations and matching dynamics is relaxed as in methods such as time-contrastive network (TCN) (Sermanet et al., 2018) and ours.

Imitation learning. Methods such as generative adversarial imitation learning (GAIL) (Ho and Ermon, 2016), use the generative adversarial network (GAN) (Goodfellow et al., 2014) framework and applies it in the context of learning an RL policy. In GAIL, the GAN's discriminator is trained with positive examples from expert trajectories and negative examples from the current policy. However, using a discriminator is only one possible way of measuring the probability of that agent's behaviour matching the expert (Abbeel and Ng, 2004; Argall et al., 2009; Finn et al., 2017a; Brown et al., 2019; Nachum and Yang, 2021). Given a vector of features, distance-based imitation learning aims to find an optimal transformation that computes a more meaningful distance between expert demonstrations and agent trajectories. Previous work has explored the area of state-based distance functions,

but most rely on the availability of an expert policy to continuously sample data (Ho and Ermon, 2016; Merel et al., 2017). In the section hereafter, we demonstrate how VIRL learns a more stable distance-based reward over sequences of images (as opposed to states) and without access to actions or expert policies.

Imitation without action data. For learning from demonstrations (LfD) problems, the goal is to replicate the behaviour of an expert π_E. GAIfO (Torabi et al., 2018b) has been proposed as an extension of GAIL that does not require data on the expert actions. However, GAIfO and other recent works in this area require access to an expert policy for sampling additional states during training (Sun et al., 2019; Yang et al., 2019). By comparison, our method can work with a single fixed demonstration example with different dynamics. Other recent work uses behavioural cloning (BC) to learn an inverse dynamics model to estimate the actions used via maximum-likelihood estimation (Torabi et al., 2018a). Still, BC often needs many expert examples and tends to suffer from state distribution mismatch issues between the *expert* policy and *student* (Ross et al., 2011a).

Additional works learn implicit models of distance that require large amounts of demonstration data and none of these explicitly learn a sequential model considering the demonstration timing (Yu et al., 2018; Finn et al., 2017b; Sermanet et al., 2018; Merel et al., 2017; Edwards et al., 2019; Sharma et al., 2019). The work in Wang et al. (2017); Li et al. (2017); Peng et al. (2019, 2021) includes a more robust GAIL framework with a new model to encode motions for few-shot imitation. However, they need access to an expert policy to sample additional data. In this work, we train recurrent siamese networks (Chopra et al., 2005) to learn more meaningful distances between videos. Other work uses state-only demonstration to out-perform the demonstration data but requires many demonstrations and ranking information to be successful (Brown et al., 2019, 2020). We show results on more complex 3D tasks and additionally model distance in time, i.e. due to the embedding of the entire sequence, our model can compute meaningful distances between agent and demonstration even if they deviate in time.

Imitation from images. Some works like Finn et al. (2017b); Sermanet et al. (2018); Liu et al. (2018); Dwibedi et al. (2018), use image-based inputs instead of states but require on the order of hundreds of demonstrations. Further, these models only address spatial alignment, matching joint positions/orientations to a single state, and can not implicitly provide an additional signal related to temporal ordering between expert demonstration and agent motion like our recurrent sequence model does. Other works that imitate image-based information do so only between goal states (Pathak* et al., 2018).

Imitation from few images with different dynamics. Comparative methods like TCN use metric learning to embed *simultaneous viewpoints* of the same object (Sermanet et al., 2018). They use TCN embeddings as *features in the system state* which are provided to PILQR (Chebotar et al., 2017) reinforcement learning algorithm, which combines model-based learning, linear time-varying dynamics and model-free corrections. In contrast, our Siamese network-based approach *learns the reward* for an arbitrary subsequent RL algorithm. Our method does not rely on multiple views, and we use a recurrent neural network (RNN)-based autoencoding approach to regularize the distance computations used for generated rewards. These choices allow VIRL to achieve performance gains over TCN, as shown in Section 5.

3. Preliminaries

In this section, we provide a very brief review of the fundamental background used by our method. reinforcement learning (RL) is formulated within the framework of a Markov decision process (MDP) where at every time step t, the world (including the agent) exists in a state $s_t \in S$, where the agent is able to perform actions $a_t \in A$. The action to take is determined according to a policy $\pi(a_t|s_t)$ which results in a new state $s_{t+1} \in S$ and reward $r_t = R(s_t, a_t, s_{t+1})$ according to the transition probability function $P(s_{t+1}|s_t, a_t)$. The policy is optimized to maximize the future discounted reward $\mathbb{E}_{r_0,\ldots,r_T}\left[\sum_{t=0}^{T} \gamma^t r_t\right]$, where T is the max time horizon, and γ is the discount factor. The formulation above generalizes to continuous states and actions, which is the situation for the agents we consider in our work.

Imitation Learning. Imitation learning is typically cast as the process of training a new policy to reproduce expert policy behaviour. Behaviour cloning is a fundamental method for imitation learning. Given an expert policy π_E possibly represented as a collection of trajectories $\tau = \langle (s_0, a_0), \ldots, (s_T, a_T) \rangle$ a new policy π can be learned to match this trajectory using supervised learning and maximizing the expectation $\mathbb{E}_{\pi_E}\left[\sum_{t=0}^{T} \log \pi(a_t|s_t, \theta_\pi)\right]$. While this simple method can work well, it often suffers from distribution mismatch issues leading to compounding errors as the learned policy deviates from the expert's behaviour (Ross et al., 2011b). Inverse reinforcement learning avoids this issue by extracting a reward function from observed optimal behaviour (Ng et al., 2000). In our approach, we learn a distance function that allows an agent to compare an observed behaviour to its current behaviour to define its reward r_t at a given time step. Our method only requires a single reference activity, but the comparison network can be trained across a collection of different behaviours. Further, we do not assume the example data to be optimal. See Appendix 7.2 for further details of the connections of our work to inverse reinforcement learning.

Variational Auto-encoders VAEs are a popular approach for learning lower-dimensional representations of a distribution (Kingma and Welling, 2014). A VAE consists of two parts, an encoder $q(\mathbf{z}|\mathbf{x}, \phi)$, with parameters ϕ and a decoder $p(\hat{\mathbf{x}}|\mathbf{z}, \psi)$ with parameters ψ. The encoder maps inputs $\mathbf{x}$ to a latent encoding $\mathbf{z}$, and, in turn, the decoder transforms $\mathbf{z}$ back to a reconstruction $\hat{\mathbf{x}}$. The model parameters for both ϕ and ψ are trained jointly to maximize

$$\mathcal{L}_{VAE}(s, \phi, \psi) = -D_{KL}(q(\mathbf{z}|\mathbf{x}, \phi)|p(\mathbf{z})) + \mathbb{E}_{q(\mathbf{z}|\mathbf{x},\phi)}[\log p(\mathbf{x}|\mathbf{z}, \psi)], \tag{1}$$

where D_{KL} is the Kullback-Leibler divergence and $p(\mathbf{z})$ is a prior distribution over the latent space. The encoder (inference model) takes the form of a multivariate diagonal covariance distribution $q(\mathbf{z}|\mathbf{x}, \phi) = \mathcal{N}(\mu(\cdot|\mathbf{x}, \phi), \sigma^2(\cdot|\mathbf{x}, \phi))$, where the mean $\mu(\cdot|\mathbf{x}, \phi)$ and variance $\sigma^2(\cdot|\mathbf{x}, \phi)$ are typically given by a deep neural network.

VAEs have been extended to work on sequences of images with the inclusion of an RNN, e.g. in Chung et al. (2015) and the notation in the following section follows the one from that work. Sequence-to-sequence models can be used to learn the conditional probability of one sequence given another $p(y_0, \ldots, y_{T'}|x_0, \ldots, x_T)$, where $\mathbf{x} = x_0, \ldots, x_T$ and $\mathbf{y} = y_0, \ldots, y_{T'}$ are sequences. Here, we will use extensions of encoder-decoder recurrent neural networks which learn a latent representation $\mathbf{h}$ that compresses the information. For VIRL we reuse the VAE encoder mean from Equation 1 to encode individual images $e_t \leftarrow \mu(x_t, \phi)$ for

which we learn the sequence encoder f, denoted as $\mathbf{h} \leftarrow f(e_0, \ldots, e_T, \omega)$ with parameters ω. Conversely, a sequence decoder g, conditioned on $\mathbf{h}$, reconstructs the original input sequence $x_0, \ldots, x_T$. First by producing a decoded sequence of the correct length $y_0, \cdots, y_t \leftarrow g(\mathbf{h}, \rho)$ with parameters ρ and second, using the decoder from Equation 1 by $\hat{\mathbf{x}}_t \leftarrow p(\mathbf{y}_t, \psi)$. The loss for decoding the original sequence $\mathbf{X} = \{x_0, \ldots, x_t\}$ can then be written as the L_2 distance

$$\mathcal{L}_{RAES}(\mathbf{X}, \phi, \psi, \omega, \rho) = ||\mathbf{X} - \{p(\mathbf{y}_0, \psi), \ldots, p(\mathbf{y}_t, \psi)\}||^2 \tag{2}$$

This method works for learning compressed representations for transfer learning (Zhu et al., 2016) and 3D shape retrieval (Zhuang et al., 2015). In our case, this type of autoencoding can help regularize our model by forcing the encoding to contain all the information needed to reconstruct the trajectory.

4. Visual Imitation with Reinforcement Learning

In this work, we create a new method for performing imitation from only visual data (no actions) and use reinforcement learning to fill in the missing action data by training the agent to find the actions that result in matching the original distribution of observations.

The Sequence Encoder/Decoder Networks Figure 2a shows an outline of the network design. There are 2 LSTMs, one for sequence encoding and one for sequence decoding, as well as one encoder CNN, and one decoder CNN, all of which are shared across the agent and expert, similar to a Siamese network. A single convolutional network $e_t^e \leftarrow \mu(\mathbf{o}_t; \phi)$ is used to transform observations (images) at every timestep t of the demonstration from the expert $\mathbf{o}_t^e$ or the agent $\mathbf{o}_t^a$ to the corresponding encoding vector e_t. After the observations are passed through the image encoder, the result is an encoded sequence $\langle e_0^e, \ldots, e_t^e \rangle$ (for the expert in this example), this sequence is fed into the `LSTM` sequence encoder until a final encoding is produced $\mathbf{h}_t^e \leftarrow f(e_0^e, \ldots, e_T^e; \omega)$. This same process is performed over the agent data $\mathbf{o}^a$ producing $\mathbf{h}_t^a \leftarrow f(e_0^a, \ldots, e_T^a; \omega)$. These sequence encodings $\mathbf{h}_t^e$ and $\mathbf{h}_t^a$ are fed into the sequence decoder $g(\mathbf{h}_t; \rho)$ separately, which generates a series of decoded latent representations $\langle \hat{y}_0^e, \ldots, \hat{y}_t^e \rangle$ which are then decoded back to images with a deconvolutional network $\hat{o} \leftarrow p(\hat{y}_t^e; \psi)$. The same process is applied to both the agent and expert using the same image encoder ϕ and decoder ψ and sequence encoder ω, and decoder ρ.

Loss Terms The siamese-loss between a fully encoded demonstration sequence h_t^e and a sequence of the agent h_t^a forces not just individual frames but the representation of entire sequences to match if they are from the same policy. This siamese network sequence loss $\mathcal{L}_{SNS}$ is defined in Eq.3. A frame-by-frame siamese-loss between e_t^e of the demonstration and e_t^a of the agent encourages individual frames to have similar encodings as well. This siamese network image loss $\mathcal{L}_{SNI}$ uses Eq.3 as well but is trained over pairs of images. We define The Siamese network loss (both for images and sequences) as:

$$\begin{aligned} \mathcal{L}_{SNX}(\mathbf{o}_i, \mathbf{o}_p, \mathbf{o}_n, y; \phi) = & y * ||f(\mathbf{o}_i; \phi) - f(\mathbf{o}_p; \phi)||^2 + \\ & (1 - y) * (\max(\rho - (||f(\mathbf{o}_i; \phi) - f(\mathbf{o}_n; \phi)||^2), 0)), \end{aligned} \tag{3}$$

where $y \in [0, 1]$ is the indicator for negative/positive samples. When $y = 1$, the pair is positive, and the distance between current observation $\mathbf{o}_i$ to positive sample $\mathbf{o}_p$ should

be minimal. When $y = 0$, the pair is negative, and the distance between $\mathbf{o}_i$ and negative example $\mathbf{o}_n$ should be increased. In subsection 4.1 the details of how the positive and negative examples are constructed is outlined. We compute this loss over batches of data that are half positive and half negative pairs. The margin ρ is set to 1 and is used as an attractor or anchor to pull the negative example output away from $\mathbf{o}_i$ and push values towards a $[0, 1]$ range. $f(\cdot)$ computes the output from the underlying network (i.e. `Conv` or `LSTM`). The data used to train the Siamese network is a combination of observation trajectories $\mathbf{O} = \langle \mathbf{o}_0, \ldots, \mathbf{o}_T \rangle$ generated from simulating the agent in the environment and the demonstration. For our recurrent model the observations $\mathbf{O}_p, \mathbf{O}_n, \mathbf{O}_i$ are sequences. We additionally train the encoding of a single observation of either agent or expert at a given timestep using the VAE loss $\mathcal{L}_{VAE}$ from Eq.1. Lastly, the entire sequence of observations of both the agent and expert is encoded and then decoded back separately, as shown in Fig. 13, and the `LSTM`*s* are trained with the loss $\mathcal{L}_{RAES}$ from Eq.2. We found using these image- and sequence-autoencoders important for improving the latent space conditioning. This combination of image-based and sequence-based losses assists in compressing the representation while ensuring intermediate representations remain informative. The combined loss to train the model is:

$$
\begin{aligned}
\mathcal{L}_{VIRL}(\mathbf{O}_i, \mathbf{O}_p, \mathbf{O}_n, y; \phi, \psi, \omega, \rho) = & \underbrace{\lambda_1 \mathcal{L}_{SNS}(\mathbf{O}_i, \mathbf{O}_p, \mathbf{O}_n, y; \phi, \omega)}_{\text{Contrastive sequence loss (Eq3)}} + \\
& \underbrace{\lambda_2 \Big[\frac{1}{T} \sum_{t=0}^{T} \mathcal{L}_{SNI}(\mathbf{O}_{i,t}, \mathbf{O}_{p,t}, \mathbf{O}_{n,t}, y; \phi) \Big]}_{\text{Contrastive frame loss (Eq3)}} + \\
& \underbrace{\lambda_3 [\mathcal{L}_{RAES}(\mathbf{O}_i; \phi, \psi, \omega, \rho) + \mathcal{L}_{RAES}(\mathbf{O}_p; \phi, \psi, \omega, \rho) + \mathcal{L}_{RAES}(\mathbf{O}_n; \phi, \psi, \omega, \rho)]}_{\text{Recurrent autoencoder loss (Eq.2)}} + \\
& \underbrace{\lambda_4 \Big[\frac{1}{T} \sum_{t=0}^{T} \big[\mathcal{L}_{VAEI}(\mathbf{O}_{i,t}; \phi, \psi) + \mathcal{L}_{VAEI}(\mathbf{O}_{p,t}; \phi, \psi) + \mathcal{L}_{VAEI}(\mathbf{O}_{n,t}; \phi, \psi) \big] \Big]}_{\text{Variational autoencoder loss (individual frames) (Eq1)}}.
\end{aligned}
\tag{4}
$$

Where the relative weights of the different terms are $\lambda_{1:4} = \{0.7, 0.1, 0.1, 0.1\}$, the image encoder convnet is ϕ, the image decoder ψ, the recurrent encoder ω, and the recurrent decoder ρ. The weights for λ are found by empirically evaluating VIRL over all environments from section 5. Additional details on the hyperparameter search can be found in subsection 7.8.

Reward Calculation The model trained using the loss function described above is used to calculate the distance between two sequences of observations seen up to time t as $d(\mathbf{O}^e, \mathbf{O}^a; \phi, \omega) = ||f(\mathbf{o}^e_{0:t}; \omega) - f(\mathbf{o}^a_{0:t}; \omega)|| + ||f(\mathbf{o}^e_t; \phi) - f(\mathbf{o}^a_t, \phi)||$ and the reward as $r(\mathbf{o}^e_{0:t}, \mathbf{o}^a_{0:t}) = -d(\mathbf{O}^e, \mathbf{O}^a; \phi, \omega)$. During RL training, we compute a distance given the sequence observed so far in the episode. The sequence-based distance can model time-invariant distances, and the image-based distance can match the expert demonstration more precisely. In subsection 5.2 we experimentally show the importance of each distance for imitation learning.

Algorithm 1 VIRL

1: Initialize parameters θ_π, $\omega, \phi, \rho, \psi D \leftarrow \{\}$
2: **while** not done **do**
3: **for** $i \in \{0, \ldots, N\}$ **do** ▷ Collect N trajectories
4: $\{s_0, \mathbf{o}_0^e, \mathbf{o}_0^a\} \leftarrow$ env.reset(), $\tau^i \leftarrow \{\}$ ▷ Env returns state and observations, no reward
5: **for** $t \in \{0, \ldots, T\}$ **do** ▷ Collect a trajectory
6: $a_t \leftarrow \pi(\cdot|s_t, \theta_\pi)$ ▷ policy produces an action given that agent state
7: $\{s_{t+1}, \mathbf{o}_{t+1}^e, \mathbf{o}_{t+1}^a\} \leftarrow$ env.step(a_t)
8: $\tau_t^i \leftarrow \{s_t, \mathbf{o}_t^e, \mathbf{o}_t^a, a_t\}$, $\mathbf{O}_{i,t}^e \leftarrow \mathbf{o}_t^e$, $\mathbf{O}_{i,t}^a \leftarrow \mathbf{o}_t^a$ ▷ store experience for later training
9: $\{s_t, \mathbf{o}_t^e, \mathbf{o}_t^a\} \leftarrow \{s_{t+1}, \mathbf{o}_{t+1}^e, \mathbf{o}_{t+1}^a\}$
10: **end for**
11: $D \leftarrow D \bigcup \{\tau^0, \ldots, \tau^N\}$ ▷ add trajectories to offline dataset
12: Perform M batch updates $d(\cdot, \cdot)$ parameters ω, ϕ, ρ, ψ using D and Equation 4
13: $\mathbf{r}_{0:T}^{0:N} \leftarrow -d(\mathbf{O}_{0:N,0:T}^e, \mathbf{O}_{0:N,0:T}^a|\omega, \phi)$ ▷ compute rewards with updated ϕ, ω
14: Update θ_π with $\{\{\tau^0, \mathbf{r}^0\}, \ldots, \{\tau^N, \mathbf{r}^N\}\}$ ▷ RL policy update.
15: **end for**
16: **end while**

Training the Model Details of the algorithm used to train the distance metric and policy are in 1. We consider a variation on the typical RL environment that produces three different outputs, two for the agent and one for the demonstration and no reward. The first is the internal robot pose and link velocities, which we refer to as the state s_t. The second and third are images of the agent, or observation $\mathbf{o}_t^a$ and the demonstration $\mathbf{o}_t^e$, shown in Figure 2b. The images are used with the distance metric to compute the similarity between the agent and the demonstration. We train the agent's policy using the trust-region policy optimization (TRPO) algorithm (Schulman et al., 2015). The policy uses the state s_t as input, which is easier to access than the 3rd person video generated by the agent during test time, and is trained online (line 15) using the learned distance function (line 14). The use of off-policy training increases the LSTM-based distance function's training efficiency. The off-policy training also allows us to train the distance function using data from other tasks to increase the robustness of the model while fine-tuning the current task, as we will describe in Section 5.2.

4.1 Unsupervised Data Labelling and Generation

To construct *positive* and *negative* pairs for training, we make use of time information and adversarial information. We use timing information where observations at similar times in the same sequence are often correlated, and observations at different times are less likely to be similar. We use these ideas to provide labels for the positive and negative pairs to train the Siamese network. Positive pairs are created by adding Gaussian noise with $\sigma = 0.05$ to the images in the sequence, duplicating, or shifting random frames of the sequences. Negative pairs are created by shuffling, cropping or reversing one sequence. VIRL will learn to decode these modified sequences which helps the model be robust to noise. Additionally, we include *adversarial* pairs where positive pairs come from the same distribution, for example, two motions for the agent or two from the demonstration at different times. Negative pairs then

include one from the expert and one from the agent. A combination of these augmentations are chosen randomly and applied to the current training batch and are not added to the replay buffer. Additional details on how the shuffling, swapping, and the use of adversarial pairs are available in the Appendix 7.7.

Data Augmentation We apply several data augmentation methods to produce additional data variation for training the distance metric. Using methods analogous to the cropping and warping methods popular in computer vision (He et al., 2015) we randomly *crop* sequences and randomly *warp* the demonstration timing. The *cropping* is performed by removing later portions of the training sequences during batch updates. These augmentations allow the distance function to learn a more general representation of motions, such as a walk even if the walking demonstration includes a single step or multiple steps at a different speed and pauses between steps. This cropping is denoted as early episode sequence priority (EESP) where the probability of cropping out a window in the sequence $\mathbf{x}$ at i is $p(i) = \frac{len(\mathbf{x})-i}{\sum i}$, increases the likelihood of cropping earlier in the sequence. As the agent improves, the average length of each episode increases, and so too will the average length of the cropped window. The motion *warping* performs a type of scaling to the time dimension of the demonstration. This is accomplished by creating a continuous function of the demonstration so we can sample the motion at different speeds $\hat{O} \leftarrow f(O, \delta)$. For example, if the motion is replayed at $\delta = 0.5$ the motion $\hat{O}$ will be twice as long as O. Linear interpolation is used to fill in information between frames. This allows for training the distance function to recognize that, for example, a walking motion with one step in it and another with two steps are still examples of a walk. Last, we use reference state initialization (RSI) (Peng et al., 2018), where we generate the initial state of the agent and expert randomly from the demonstration. With this property, the environment functions as a form of memory replay. The environment allows the agent to go back to random points in the demonstration as if replaying a remembered demonstration and collect new data from that point in the demonstration. The experiments in Sec. 5.2 show the importance of these augmentation methods in terms of improving the robustness of the learned comparator and resulting policy.

5. Experiments, Results and Analysis

We evaluate VIRL compared to previous methods in terms of sample efficiency and task-solving capability. The comparison is over a collection of different simulation environments. In these simulated robotics environments, we task the agent with imitating a given reference video demonstration. Each simulation environment provides a hard-coded reward function based on the agent's pose to evaluate the policy quality independently. The imitation environments include challenging and dynamic tasks for humanoid, dog and raptor robots. Some example tasks are running, jumping, trotting, and walking, shown in Fig. 3 and Fig. 4. The demonstration $\mathbf{o}_t^e$ the agent uses for visual imitation learning is produced from a clip of motion capture data for each task. The motion capture data animates a kinematically controlled robot in the simulation for capturing video. Because the demonstration is kinematically generated, the agent also needs to learn how to bridge the gap between the different dynamics in the demonstration that may be impossible to reproduce exactly. We convert the images captured from the simulation to 48×48 grey-scale images. Third-person

image data is often not available during test time; therefore, the agent's policy instead receives the environment state as the link distances and velocities relative to the robot's centre of mass (COM).

2D Imitation Tasks The first group of evaluation environments contain a set of agents with different morphologies. In Figure 3 we show images from the 2D humanoid, dog and raptor environment. In these environments the rendering and simulation is in 2D, reducing the complexity of the control system and dynamics, and allowing for faster training times.

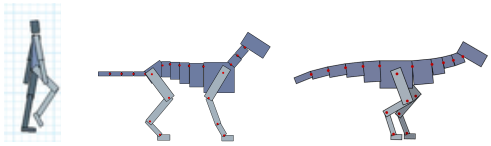


Figure 3: Frames from the humanoid2d, dog2d, and raptor2d environments used in the experiments.

3D Imitation Tasks For further evaluation, we compare the performance on 3D humanoid and two quadrupedal robot simulators used for Sim2Real research, the Laikago (Peng et al., 2020) and Pupper (Kau et al., 2020). The humanoid3d environment has multiple tasks that can be solved and used to generate data from other tasks for training the distance metric. These tasks include: walking, running, jogging, front-flips, back-flips, dancing, jumping, punching and kicking). To perform the data augmentation described in Section 4.1 we also construct data from a modified version of each task with a randomly generated playback speed modifier $\delta \in [0.5, 2.0]$ e.g. walking-dynamic-speed, which warps the demonstration timing. This additional data provides a richer understanding of distances in space and time with the distance metric. As we will show later in this section, VIRL learns policies that produce similar behaviour to the demonstration across these diverse tasks. We show example trajectories from the learned policies in Fig. 4 and in the supplemental Video. It takes $5-7$ days to train each policy in these results on a 16 core machine with an Nvidia GTX1080 GPU.

Comparison Methods We compare VIRL to two baselines that learn distances in observation space. The first is GAIfO (Torabi et al., 2018b) that trains a GAN to differentiate between images from the demonstration and images from the agent. The other is TCN, an image-to-image only siamese model (Nair et al., 2018). These methods have been used to

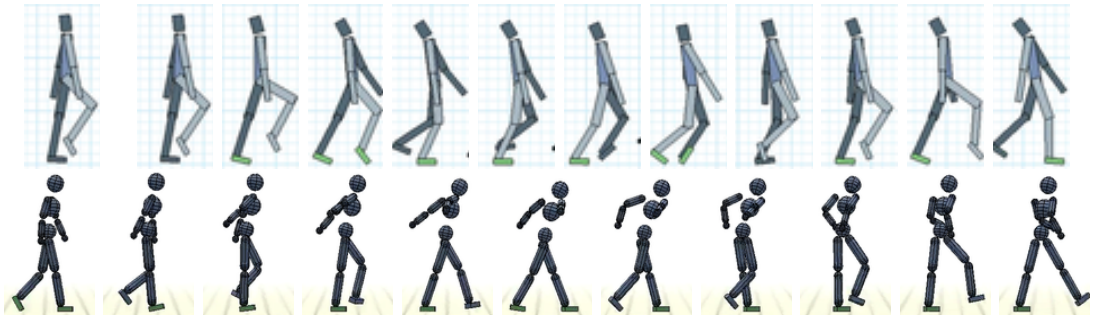
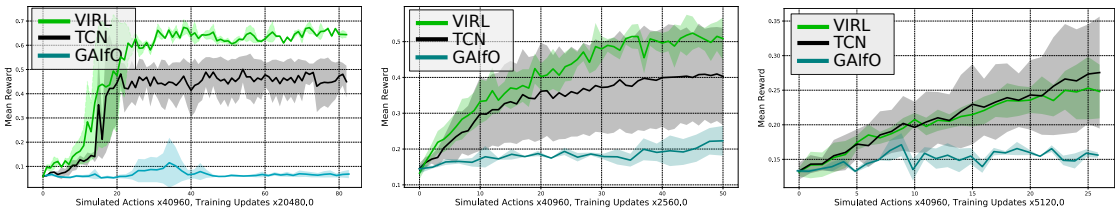


Figure 4: Frames of the agent's motion after training on humanoid2d and humanoid3d walking. Additionally, the dog, raptor, zombie walk, run and jumping policy can be found on the project website: https://sites.google.com/view/virl1.



(a) humanoid2d walk (b) dog2d (c) raptor2d

Figure 5: Comparisons of VIRL, TCN and GAIfO with (a) the humanoid2d, (b) the 2D dog agent, and (c) the 2D raptor agent. GAIfO struggles to show improvement on these tasks. TCN does make progress on these imitation tasks but the performance is not as good as VIRL. The results show average performance over 4 randomly seeded policy training simulations.

perform types of imitation from observation before. However, as we will see, they either require a significant amount of data to train or result in lower-quality reward functions and, as a result, lower-quality policies.

5.1 Learning Performance

In Figure 5 we present results across different agent types including the 2D walking humanoid in Section 5a, a 2D trotting dog in Fig. 5b, and a 2D Raptor in Fig. 5c. We compare VIRL directly with both GAIfO and TCN – the strongest comparable prior method. VIRL is able to provide a denser reward signal due to the LSTM-based distance that can express similarity between motion styles. While TCN is simpler, not using an LSTM model, it does not provide as rich of a reward signal for the agent to learn from. As a result, the learned reward can often be more sparse, slowing down learning. Similarly, GAIfO has difficulty learning a robust and smooth distance function with the little demonstration data available. This leads to very jerky motion or agent's that stand still, matching the average pose of the demonstration, both of which are contained in the demonstration distribution but do not capture sequential-temporal-behaviour well.

In Figure 6 we compare VIRL with TCN, across many different and more challenging humanoid3d tasks in Fig. 6(a-d). Across these experiments, we observe that VIRL learns faster and produces higher value policies. In particular, we find that VIRL does very well compared to TCNs, which represents the strongest prior approach capable of performing this task of which we are aware. The humanoid3d tasks are particularly challenging as they have high control dimensionality causing the agent to deviate from the desired imitation behaviour easily. These tasks also contain higher levels of partial observability compared to the 2D experiments. In these environments, the temporal distance in VIRL provides a crucial additional reward signal that helps the agent match the style of the motion early on in training despite the partial information observations.

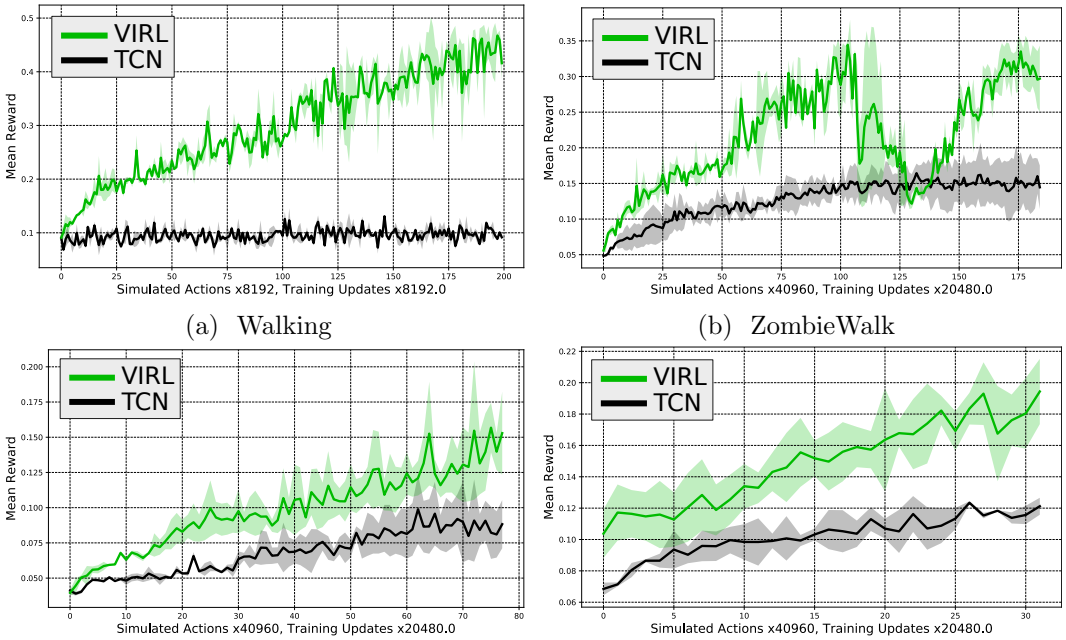
5.2 Analysis and Ablation

Sequence Encoding Using the learned sequence encoder, we compute the encodings across a collection of different motions and create a t-distributed stochastic neighbor embedding

(t-SNE) embedding of the encodings (Maaten and Hinton, 2008). In Fig. 7a we plot motions both generated from the learned policy π and the expert trajectories π_E. Overlaps in specific areas of the space for similar classes across learned π and expert π_E data indicate a well-formed distance metric that does not separate expert and agent examples. There is also a separation between motion classes in the data, and the cyclic nature of the walking cycle is visible.

Ablation In Fig. 7b we compare the importance of the spatial distance $||f(\mathbf{o}_t^e;\phi) - f(\mathbf{o}_t^a,\phi)||^2$ using the image encoder ϕ and temporal-LSTM-distance $||f(\mathbf{o}_{0:t}^e;\omega) - f(\mathbf{o}_{0:t}^a;\omega)||^2$ using the sequence encoder ω of VIRL. Using the recurrent representation alone (LSTM only) allows learning to progress quickly but can lead to difficulties in informing the policy on how to match the desired demonstration more precisely. On the other hand, using only the encoding between single frames as is done with TCN, slows learning due to little reward when the agent quickly becomes out-of-sync with the demonstration behaviour. The best result is achieved by combining the representations from these two models (VIRL).

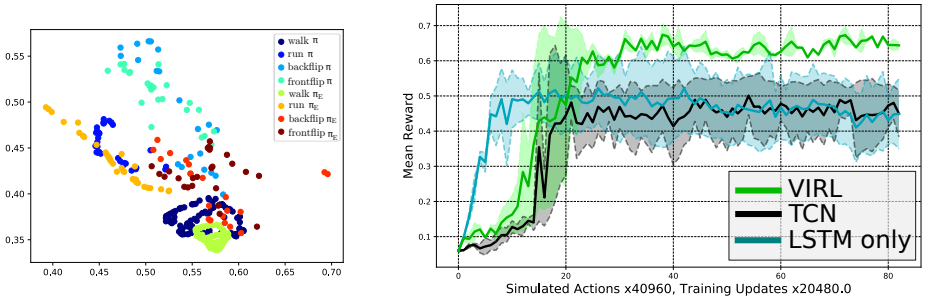
Data augmentation comparisons. We conduct ablation studies for learning policies for 3D humanoid control in Fig. 8a and 8b. We compare the effects of data augmentation methods, network models and the use of additional data from other tasks to train the siamese network (24 additional tasks such as back-flips, see appendix 7.3 for more details on these tasks). We also compared using different length sequences for training, shorter (where the probability of the length decays linearly), uniform random and max length available. For these more complex and challenging three-dimensional humanoids (humanoid3d) control



(a) Walking (b) ZombieWalk

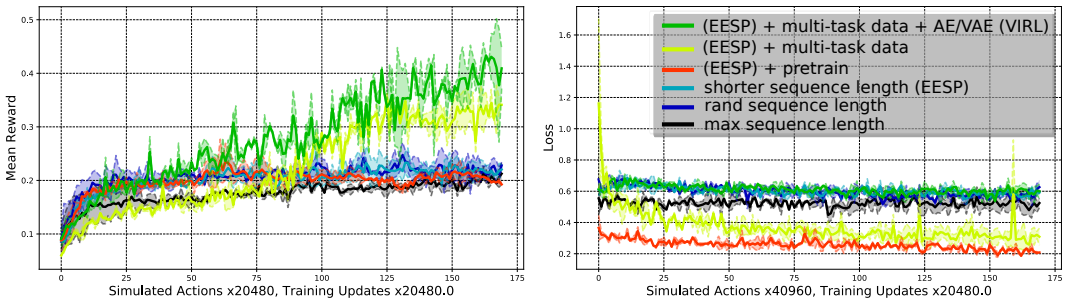
(c) Running (d) Jumping

Figure 6: 3D Humanoid motion imitation experiments: Comparisons of VIRL with TCNs. These results show the average performance over 4 randomly seeded policy training simulations.



(a) t-SNE embedding (humanoid3d) (b) Distance function ablation

Figure 7: (a) t-SNE embedding that shows VIRL's learned distance landscape. (b) Comparison between combining different learned distance models. The combination of spatial and temporal distances leads to the best result (VIRL).



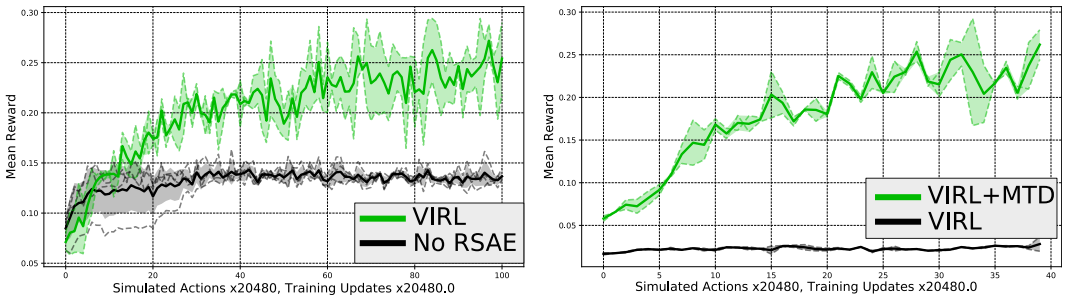
(a) 3D Walking Ablation Analysis (b) Distance Metrics

Figure 8: (a) Ablation analysis of VIRL on the humanoid3D environment showing the mean reward over 5 random seeds. The legend is the same as (b), where we examine the impact on our loss under the different distance metrics resulting from the ablation analysis. We find that including multi-task data (only available for the humanoid3D) and both the VAE and recurrent AE losses provide the most high-performing models.

problems, the data augmentation methods, including EESP, increase average policy quality marginally compared to the importance of using multi-task data, this is likely related to the increased partial observably of these tasks. However, the addition of the auto-encoding losses to VIRL results in the quickest learning and highest value policies.

The experiment in Fig. 9a highlights the improvement to VIRL afforded by the recurrent sequence autoencoder (RSAE) model component from Eq. 4 that forces the encoding to contain enough information to decode the video sequence. While the experiment in Fig. 9b shows the dramatic improvement achieved when we include offline multi-task data for training the distance function. These methods are combined together to provide substantial learning performance increases across environments for VIRL. Further analysis is available in the Appendix, including additional comparison with TCN in Fig. 15(a-b) and details on training the distance model.

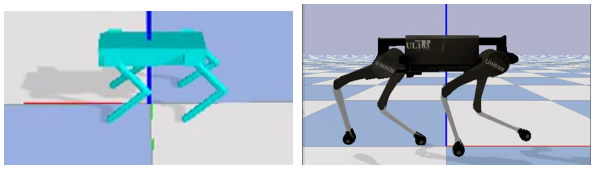
Sim2Real for Quadruped Robots We use VIRL to train policies for two simulated quadrupeds shown in Fig. 10. These environments have been used for Sim2Real transfer(Tan et al., 2018; Peng et al., 2020). The resulting behaviours learned in simulation are available



(a) ZombieWalk, LSTM (b) Running, MultiTask

Figure 9: (a) Ablating the recurrent autoencoder from VIRL (i.e. No RSAE in the figure) impairs the ability to learn how to walk like a Zombie. (b) Here, further we see how multi-task training data helps learn better policies for running (away from Zombies if desired).

Figure 10: Pupper and Laikago agents, both of which are difficult platforms to learn policies on since they tumble easily and expert demonstrations are hard to obtain.



at: https://sites.google.com/view/virl1. We find that the Laikago environment is particularly challenging to learn; however, we can learn good policies on the smaller Stanford pupper in a day. This shows that VIRL can potentially be used to learn a control policy from a single video clip and transfer that policy to real hardware, however, we leave the details of the transfer process to future work.

6. Discussion and Conclusion

In this work, we have created a new method for learning imitative policies from a single demonstration. The method uses a Siamese recurrent network to learn a distance function in both space and time. This distance function learns to imitate video data where the agent's observed state can be noisy and partially observed. We use this model to provide rewards for training an RL policy. By using data from other motion styles and regularization terms, VIRL produces policies that demonstrate similar behaviour to the demonstration in complex 3D control tasks. We found that the recurrent distance learned by VIRL was particularly beneficial when imitating demonstrations with greater partial observability.

One might expect that the distance metric should be pretrained to quickly understand the difference between a good and bad demonstration. However, we have found that in this setting, learning too quickly can destabilize learning, as rewards can change, which can cause the agent to diverge off to an unrecoverable policy space. In this setting, slower is better, as the distance metric may not yet be accurate. However, the learned distance function may be locally or relatively reasonable, which is enough to learn an acceptable policy. As learning continues, these two optimizations can converge together.

When comparing our method to GAIfO, we have found that GAIfO has limited temporal consistency. GAIfO led to learning jerky and overactive policies. The use of a recurrent discriminator for GAIfO, similar to our use of sequence-based distance, may mitigate some

of these issues and is left for future work. It is challenging to produce results better than the carefully manually crafted reward functions used by the RL simulation environments that include motion phase information in the observations (Peng et al., 2018, 2017). However, we have shown that our method can compute distances in space and time and learns faster than current methods that can be used in this domain. A combination of beginning learning with our method and following with a manually crafted reward function could potentially lead to faster learning of high-quality policies if true state information is available. Still, as environments become increasingly more complex and real-world training becomes an efficient option, methods that can learn to imitate from only video demonstration enable access to a vast collection of motion information, extensive studies on multi-task learning, and give way to more natural methods to provide instructions to agents.

Acknowledgements and Disclosure of Funding We would like to acknowledge funding support from NSERC, CIFAR, and ElementAI/Service Now and compute support from ComputeCanada and ElementAI/Service Now.

References

Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 1–, New York, NY, USA, 2004. ACM. ISBN 1-58113-838-5. doi: 10.1145/1015330.1015430. URL http://doi.acm.org/10.1145/1015330.1015430.

Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469 – 483, 2009. ISSN 0921-8890. doi: https://doi.org/10.1016/j.robot.2008.10.024. URL http://www.sciencedirect.com/science/article/pii/S0921889008001772.

Sarah-Jayne Blakemore and Jean Decety. From the perception of action to the understanding of intention. *Nature reviews neuroscience*, 2(8):561–567, 2001.

Daniel Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *International Conference on Machine Learning*, 2019.

Daniel S Brown, Wonjoon Goo, and Scott Niekum. Better-than-demonstrator imitation learning via automatically-ranked demonstrations. In *Conference on Robot Learning*, pages 330–359, 2020.

Yevgen Chebotar, Karol Hausman, Marvin Zhang, Gaurav Sukhatme, Stefan Schaal, and Sergey Levine. Combining model-based and model-free updates for trajectory-centric reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 703–711, 2017.

Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 539–546. IEEE, 2005.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. *Advances in neural information processing systems*, 28, 2015.

Debidatta Dwibedi, Jonathan Tompson, Corey Lynch, and Pierre Sermanet. Learning actionable representations from visual observations. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1577–1584, 2018. doi: 10.1109/IROS.2018.8593951.

Ashley Edwards, Himanshu Sahni, Yannick Schroecker, and Charles Isbell. Imitating latent policies from observation. In *International Conference on Machine Learning*, pages 1755–1763, 2019.

Chelsea Finn, Tianhe Yu, Justin Fu, Pieter Abbeel, and Sergey Levine. Generalizing skills with semi-supervised reinforcement learning. In *International Conference on Learning Representations*, 2017a. URL https://openreview.net/forum?id=ryHlUtqge.

Chelsea Finn, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot visual imitation learning via meta-learning. In Sergey Levine, Vincent Vanhoucke, and Ken Goldberg, editors, *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pages 357–368. PMLR, 13–15 Nov 2017b. URL https://proceedings.mlr.press/v78/finn17a.html.

Michael Gleicher. Retargetting motion to new characters. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 33–42, 1998.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1904–1916, Sept 2015. ISSN 0162-8828. doi: 10.1109/TPAMI.2015.2389824.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4565–4573. Curran Associates, Inc., 2016. URL http://papers.nips.cc/paper/6391-generative-adversarial-imitation-learning.pdf.

Nathan Kau, Aaron Schultz, Tarun Punnoose, Laura Lee, and Zac Manchester. Woofer and pupper: Low-cost open-source quadrupeds for research and education. 2020.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *International Conference onLearning Representations (ICLR)*, 2014.

Jehee Lee, Jinxiang Chai, Paul SA Reitsma, Jessica K Hodgins, and Nancy S Pollard. Interactive control of avatars animated with human motion data. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 491–500, 2002.

Yunzhu Li, Jiaming Song, and Stefano Ermon. Infogail: Interpretable imitation learning from visual demonstrations. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3812–3822. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/6971-infogail-interpretable-imitation-learning-from-visual-demonstrations.pdf.

YuXuan Liu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Imitation from observation: Learning to imitate behaviors from raw video via context translation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, page 1118–1125. IEEE Press, 2018. doi: 10.1109/ICRA.2018.8462901. URL https://doi.org/10.1109/ICRA.2018.8462901.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

Josh Merel, Yuval Tassa, Dhruva TB, Sriram Srinivasan, Jay Lemmon, Ziyu Wang, Greg Wayne, and Nicolas Heess. Learning human behaviors from motion capture by adversarial imitation. *CoRR*, abs/1707.02201, 2017. URL http://arxiv.org/abs/1707.02201.

Ofir Nachum and Mengjiao Yang. Provable representation learning for imitation with contrastive fourier features. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 30100–30112. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/fd00d3474e495e7b6d5f9f575b2d7ec4-Paper.pdf.

Ashvin V Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/7ec69dd44416c46745f6edd947b470cd-Paper.pdf.

Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2, 2000.

Deepak Pathak*, Parsa Mahmoudieh*, Michael Luo*, Pulkit Agrawal*, Dian Chen, Fred Shentu, Evan Shelhamer, Jitendra Malik, Alexei A. Efros, and Trevor Darrell. Zero-shot visual imitation. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=BkisuzWRW.

Xue Bin Peng and Michiel van de Panne. Learning locomotion skills using deeprl: Does the choice of action space matter? In *Proceedings of the ACM SIGGRAPH / Eurographics*

Symposium on Computer Animation, SCA '17, pages 12:1–12:13, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-5091-4. doi: 10.1145/3099564.3099567. URL http://doi.acm.org/10.1145/3099564.3099567.

Xue Bin Peng, Glen Berseth, Kangkang Yin, and Michiel Van De Panne. Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning. *ACM Trans. Graph.*, 36(4):41:1–41:13, July 2017. ISSN 0730-0301. doi: 10.1145/3072959.3073602. URL http://doi.acm.org/10.1145/3072959.3073602.

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Trans. Graph.*, 37(4):143:1–143:14, July 2018. ISSN 0730-0301. doi: 10.1145/3197517.3201311. URL http://doi.acm.org/10.1145/3197517.3201311.

Xue Bin Peng, Angjoo Kanazawa, Sam Toyer, Pieter Abbeel, and Sergey Levine. Variational discriminator bottleneck: Improving imitation learning, inverse RL, and GANs by constraining information flow. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HyxPx3R9tm.

Xue Bin Peng, Erwin Coumans, Tingnan Zhang, Tsang-Wei Edward Lee, Jie Tan, and Sergey Levine. Learning agile robotic locomotion skills by imitating animals. In *Robotics: Science and Systems*, 07 2020. doi: 10.15607/RSS.2020.XVI.064.

Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. Amp: Adversarial motion priors for stylized physics-based character control. *ACM Trans. Graph.*, 40(4), July 2021. doi: 10.1145/3450626.3459670. URL http://doi.acm.org/10.1145/3450626.3459670.

Rómer Rosales and Stan Sclaroff. Learning and synthesizing human body motion and posture. In *Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition (Cat. No. PR00580)*, pages 506–511. IEEE, 2000.

Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 627–635, Fort Lauderdale, FL, USA, 11–13 Apr 2011a. PMLR. URL http://proceedings.mlr.press/v15/ross11a.html.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635, 2011b.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897, 2015.

Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. Time-contrastive networks: Self-supervised learning

from video. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1134–1141, 2018. doi: 10.1109/ICRA.2018.8462891.

Pratyusha Sharma, Deepak Pathak, and Abhinav Gupta. Third-person visual imitation learning via decoupled hierarchical controller. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dÁlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 2597–2607. Curran Associates, Inc., 2019. URL http://papers.nips.cc/paper/8528-third-person-visual-imitation-learning-via-decoupled-hierarchical-controller.pdf.

Wen Sun, Anirudh Vemula, Byron Boots, and Drew Bagnell. Provably efficient imitation learning from observation alone. In *ICML*, pages 6036–6045, 2019. URL http://proceedings.mlr.press/v97/sun19b.html.

Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. In *Robotics: Science and Systems*, 2018.

Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, page 4950–4957. AAAI Press, 2018a. ISBN 9780999241127.

Faraz Torabi, Garrett Warnell, and Peter Stone. Generative adversarial imitation from observation. *arXiv preprint arXiv:1807.06158*, 2018b.

Ziyu Wang, Josh S Merel, Scott E Reed, Nando de Freitas, Gregory Wayne, and Nicolas Heess. Robust imitation of diverse behaviors. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5320–5329. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7116-robust-imitation-of-diverse-behaviors.pdf.

Chao Yang, Xiaojian Ma, Wenbing Huang, Fuchun Sun, Huaping Liu, Junzhou Huang, and Chuang Gan. Imitation learning from observations by minimizing inverse dynamics disagreement. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dÁlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 239–249. Curran Associates, Inc., 2019. URL http://papers.nips.cc/paper/8317-imitation-learning-from-observations-by-minimizing-inverse-dynamics-disagreement.pdf.

Tianhe Yu, Chelsea Finn, Annie Xie, Sudeep Dasari, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot imitation from observing humans via domain-adaptive meta-learning. *ICLR 2018 Workshop Submission*, abs/1802.01557, 2018. URL http://arxiv.org/abs/1802.01557.

Zhuotun Zhu, Xinggang Wang, Song Bai, Cong Yao, and Xiang Bai. Deep learning representation using autoencoder for 3d shape retrieval. *Neurocomputing*, 204:41–50, 2016.

Fuzhen Zhuang, Xiaohu Cheng, Ping Luo, Sinno Jialin Pan, and Qing He. Supervised representation learning: Transfer learning with deep autoencoders. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*, AAAI'08, pages 1433–1438. AAAI Press, 2008. ISBN 978-1-57735-368-3. URL http://dl.acm.org/citation.cfm?id=1620270.1620297.

7. Appendix

This section includes additional details related to VIRL.

7.1 Imitation Learning

Imitation learning is the process of training a new policy to reproduce the behaviour of some expert policy. BC is a fundamental method for imitation learning. Given an expert policy π_E possibly represented as a collection of trajectories $\tau < (s_0, a_0), \ldots, (s_T, a_T) >$ a new policy π can be learned to match these trajectory using supervised learning.

$$\max_{\theta} \mathbb{E}_{\pi_E}[\sum_{t=0}^{T} \log \pi(a_t|s_t, \theta_\pi)] \tag{5}$$

While this simple method can work well, it often suffers from distribution mismatch issues leading to compounding errors as the learned policy deviates from the expert's behaviour during test time.

7.2 Inverse Reinforcement Learning

Similar to BC, Inverse Reinforcement Learning (inverse reinforcement learning (IRL)) also learns to replicate some desired, potentially expert, behaviour. However, IRL uses the RL environment to learn a reward function that learns to tell the difference between the agent's behaviour and the example data. Here we describe maximal entropy IRL (Ziebart et al., 2008). Given an expert trajectory $\tau < (s_0, a_0), \ldots, (s_T, a_T) >$ a policy π can be trained to produce similar trajectories by discovering a distance metric between the expert trajectory and trajectories produced by the policy π.

$$\max_{c \in C} \min_{\pi} (\mathbb{E}_{\pi}[c(s, a)] - H(\pi)) - \mathbb{E}_{\pi_E}[c(s, a)] \tag{6}$$

where c is a learned cost function and $H(\pi)$ is a causal entropy term. π_E is the expert policy that is represented by a collection of trajectories. IRL is searching for a cost function c that is low for the expert π_E and high for other policies. Then, a policy can be optimized by maximizing the reward function $r_t = -c(s_t, a_t)$.

7.3 Data

We are using the mocap data from the "CMU Graphics Lab Motion Capture Database" from 2002 (http://mocap.cs.cmu.edu/). To be thorough, we provide the processing at length. This data has been preprocessed to map the mocap markers to a human skeleton. Each recording contains the positions and orientations of the different joints of a human skeleton and can therefore directly be used to animate a simulated humanoid mesh. This is a standard approach that has been widely used in prior literature (Gleicher, 1998; Rosales and Sclaroff, 2000; Lee et al., 2002). To be precise: at each mocap frame, the joints of a humanoid mesh model are set to the positions and orientations of their respective values in the recording. If a full humanoid mesh is not available, it is possible to add capsule mesh primitives between each recorded joint. This 3D mesh model is then rendered to an image through a 3rd person camera that follows the center of mass of the mesh at a fixed distance.

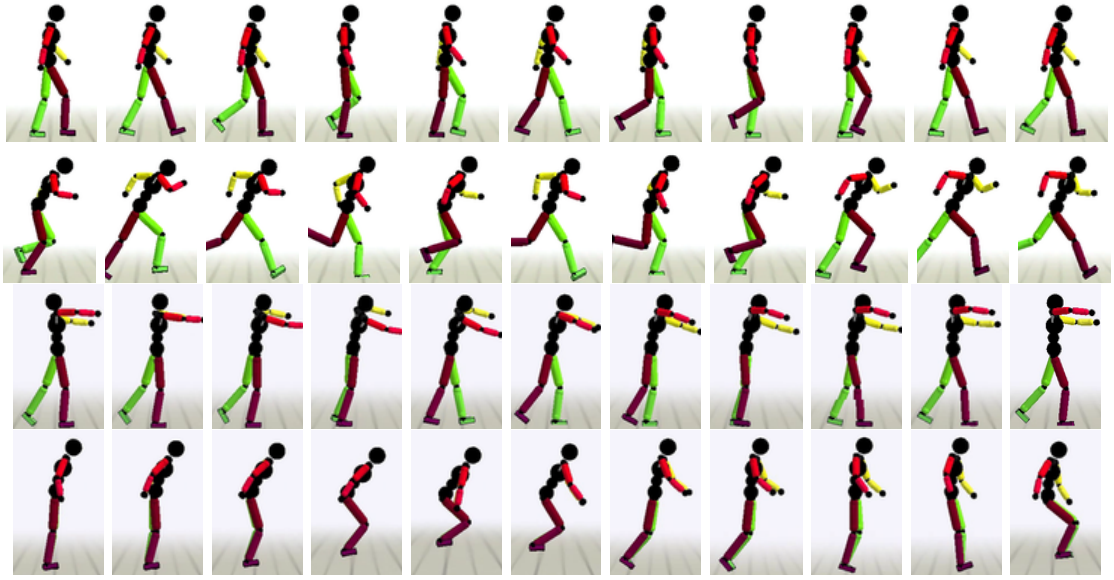


Figure 11: Rasterized frames of the imitation motions on humanoid3d walking (row 1), running (row 2), zombie (row 3) and jumping(row 4). https://sites.google.com/view/virl1

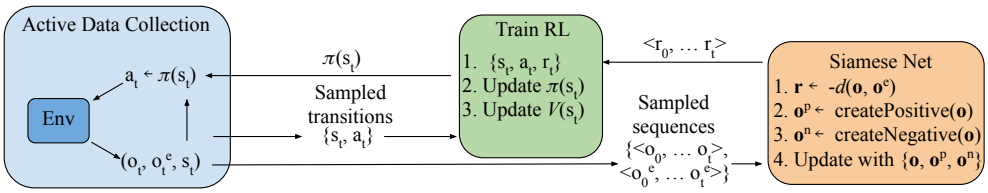


Figure 12: The flow of control for the learning system.

For the humanoid experiments, imitation data for 24 other tasks was used to help condition the distance metric learning process. These include motion clips for running, backflips, frontflips, dancing, punching, kicking and jumping along with the desired motion. The improvement due to these additional unsupervised training data generation mechanisms are shown in Fig. 8a.

The Sim2Real environments do not include video demonstrations. To create video data we use a similar method as in the other simulations. The available motion capture data is used in the simulation to control a *kinematic* character from which 3rd person video data of that agent is collected.

7.4 Training Details

The learning simulations are trained using graphics processing unit (GPU)s. The simulation is not only simulating the interaction physics of the world but also rendering the simulation scene to capture video observations. On average, it takes 3 days to execute a single training simulation. The rendering process and copying the images from the GPU is one of the most

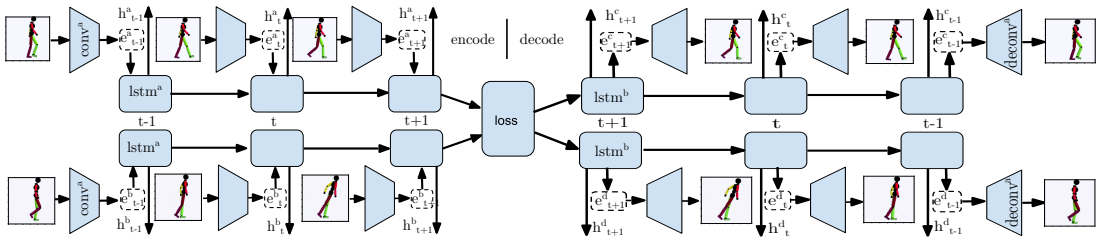


Figure 13: We use a Siamese autoencoding network structure that can provide a reward signal at every step to a reinforcement learning algorithm. For the Humanoid experiments here, the convolutional portion of our network includes 2 convolution layers of 8 filters with size 6×6 and stride 2×2, 16 filters of size 4×4 and stride 2×2. The features are then flattened and followed by two dense layers of 256 and 64 units. The majority of the network uses rectified linear unit (ReLU) activations except for the last layer that uses a sigmoid activation. Dropout is used between convolutional layers. The RNN-based model uses a LSTM layer with 128 hidden units, followed by a dense layer of 64 units. The decoder model has the same structure in reverse, with deconvolution in place of convolutional layers.

expensive operations with VIRL. We collect 2048 samples between training rounds. The batch size for TRPO is 2048. The kl term is 0.5.

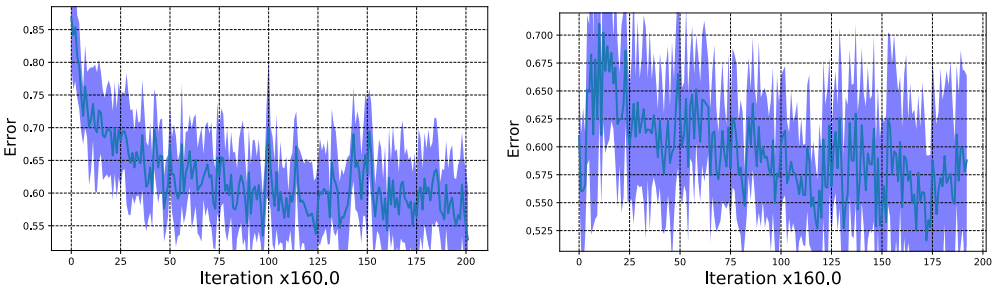
The simulation environment includes several different tasks represented by a collection of motion capture clips to imitate. These tasks come from the tasks created in DeepMimic (Peng et al., 2018). We include all humanoid tasks in this dataset. The simulation uses RSI to randomly sample start states for the agent and expert to begin. This works by first uniformly randomly selecting a time in the expert demonstration and then synchronizing the learning agent with that time in the expert demonstration. This has shown to be very helpful across many prior papers to boost learning and is used for all algorithms in this paper.

In Alg. 1 we include an outline of the algorithm used for the method and a diagram in Fig. 12. The simulation environment produces three types of observations, s_{t+1} the agent's proprioceptive pose, $\mathbf{o}^a_{t+1}$ the image observation of the agent and $\mathbf{o}^e_{t+1}$ the image-based observation of the expert demonstration. The images are grayscale 48×48. In Figure 13 we show a diagram of the network model using example data from the the humanoid3d walking task. Different network structures were evaluated, this structure with the loss defined in Equation 4 provided the best performance.

7.5 Distance Function Training

In Fig. 14a, the learning curve for the sequence-based Siamese network is shown during a pretraining phase. We can see the overfitting portion the occurs during RL training. This overfitting can lead to poor reward prediction during the early phase of training. In Fig. 14b, we show the training curve for the recurrent Siamese network after starting training during RL. After an initial distribution adaptation, the model learns smoothly, considering that the training data used is continually changing as the RL agent explores.

It can be challenging to train a sequenced-based distance function. One particular challenge is training the distance function to be accurate across the space of possible states. We found that a good strategy was to focus on the earlier parts of the episode. When the model is not accurate on states earlier in the episode, it may never learn how to get into



(a) Siamese loss during pretraining (b) Siamese loss after pretraining

Figure 14: Training losses for the Siamese distance metric.

good states later, even if the distance function understands those better. Therefore, when constructing batches to train the RNN on, we give a higher probability of starting earlier in episodes EESP. We also give a higher probability of shorter sequences as a function of the average episode length. As the agent gets better average episodes length increases, so to will the randomly selected sequence windows.

We found in our experiments that keeping the same *in-order* sequence for decoding forced the encoding model to encode long-term temporal information from the beginning of training. This is particularly challenging to cope with as the RL policy is exploring different state distributions, further exasperating the challenging problem of learning good temporal representations. Instead, we reverse the decoding sequence which allows the training model to pickup on shorter term temporal dependencies quickly. This shorter but **more consistent** representation provides better signal to the RL agent.

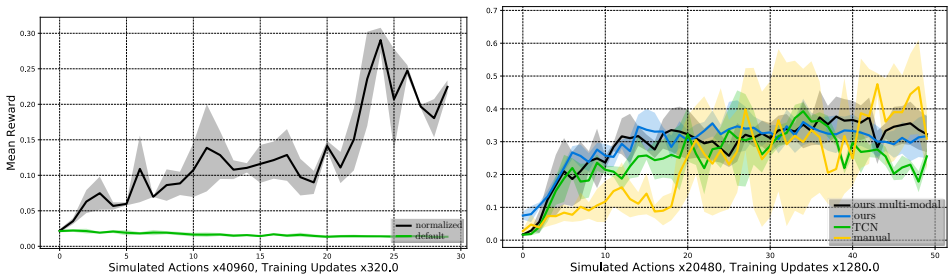
7.6 Distance Function Use

We find it helpful to *normalize* the distance metric outputs using $r = \exp(r^2 * w_d)$ where $w_d = -5.0$ scales the filtering width. This normalization is a standard method to convert distance-based rewards to be positive, which makes it easier to handle episodes that terminate early (Peng and van de Panne, 2017; Peng et al., 2018, 2019). Early in training, the distance metric often produces large, noisy values. The RL method regularly tracks reward scaling statistics; the initial high variance data reduces the significance of better distance metric values produced later on by scaling them to small numbers. The improvement of using this normalized reward is shown in Fig. 15a. In Fig. 15b we compare to a few baseline methods. The *manual* version uses a carefully engineered reward function from (Peng et al., 2017).

7.7 Positive and Negative Examples

We use two methods to generate positive and negative examples. The first method is similar to TCN, where we can assume that sequences that overlap more in time are more similar. We generate two sequences for each episode, one for the agent and one for the imitation motion. Here we list the methods used to alter sequences for positive pairs.

1. Adding Gaussian noise to each state in the sequence (mean = 0 and variance = 0.02)



(a) Reward smoothing

(b) Comparisons with other reward methods on humanoid2d

Figure 15: Ablation analysis of VIRL. We find that training RL policies is sensitive to the size and distribution of rewards. We compare VIRL to several baselines.

2. Out of sync versions where the first state from the first and the last ones from the second sequence are removed.

3. Duplicating the first state in either sequence

4. Duplicating the last state in either sequence

We alter sequences for negative pairs by

1. Reversing the ordering of the second sequence in the pair.

2. Randomly picking a state out of the second sequence and replicating it to be as long as the first sequence.

3. Randomly shuffling one sequence.

4. Randomly shuffling both sequences.

5. Using one sequence from the expert and one from the agent. We call these adversarial sequences pairs.

6. In the examples that include additional motion classes, the negatives are selected from the other classes.

The second method we use to create positive and negative examples is by including data for additional classes of motion. These classes denote different task types. For the humanoid3d environment, we generate data for walking-dynamic-speed, running, backflipping and front-flipping. Pairs from the same tasks are labelled as positive, and pairs from different classes are negative.

Viewpoint Invariance Because we perform many types of data augmentations, clipping, warping, cropping, etc, the video data can be collected from viewpoints with different angles and distances, as long as most of the agent is captured by the video. The data augmentations are designed to help increase the diversity of data, but they also result in VIRL being able to compute distances from noisy data from different view locations.

7.8 Hyper Parameter Analysis

To determine the best values for $\lambda_{1:4} = \{0.7, 0.1, 0.1, 0.1\}$ in Equation 4 we perform grid search over possible values in 0.1 increments where $\sum_{i=0}^{4} \lambda_i = 1$. This evaluation is performed over all tasks using 3 seeds on each task in section 5 and the parameters that results in the best learning performance are selected. The final parameters are designed to be robust to new environments and may not require additional tuning.