Noise-conditioned Energy-based Annealed Rewards (NEAR)

A Generative Framework for Imitation Learning from Observation

by

Anish Abhijit Diwan

In partial fulfilment of the requirements for the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday September 19, 2024 at 9:00 AM.

Student Name:	Anish Abhijit Diwan (5706580)
Programme:	MSc. Robotics
Department & Faculty:	Cognitive Robotics, Mechanical Engineering
Project duration:	Jan 2024 – Sep 2024
Supervisors:	Dr. Jens Kober [1] Dr. Julen Urain [2] Dr. Jan Peters [2]
Thesis committee:	Dr. Jens Kober Dr. Julen Urain Dr. J. Micah Prendergast [1] Dr. P. Mohajerin Esfahani [3]

Cognitive Robotics, TU Delft [1] Intelligent Autonomous Systems, TU Darmstadt [2] Delft Center for Systems and Control, TU Delft [3]

TU Delft

Noise-conditioned Energy-based Annealed Rewards (NEAR): A Generative Framework for Imitation Learning from Observation

Anish Abhijit Diwan

Abstract

This paper introduces a new imitation learning framework based on energy-based generative models capable of generating complex, life-like, physics-dependent motions, through state-only expert motion trajectories. Our algorithm, called Noise-conditioned Energy-based Annealed Rewards (NEAR), constructs several perturbed versions of the expert's motion data distribution and learns smooth, and well-defined representations of the data distribution's energy function using denoising score matching. We propose to use these learnt energy functions as reward functions to learn imitation policies via reinforcement learning. We also present a strategy to gradually switch between the learnt energy functions, ensuring that the learnt rewards are always well-defined in the manifold of policy-generated samples, thereby improving the learnt policies. We evaluate our algorithm on complex humanoid tasks such as locomotion and martial arts and compare it with state-only adversarial imitation learning algorithms like Adversarial Motion Priors (AMP). Our framework sidesteps the optimisation challenges of conventional generative imitation learning techniques and produces results comparable to AMP in several quantitative metrics across multiple tasks. Finally, a portion of this paper also analyses the optimisation challenges of adversarial imitation learning algorithms, and discusses some previously under-explored failure modes, providing rigorous empirical results to back our argumentation. Code and videos are available at anishhdiwan.github.io/noise-conditioned-energy-based-annealed-rewards/.

1 Introduction

Learning skills through imitation is probably the most cardinal form of learning for human beings. Whether it is a child learning to tie their shoelaces, a dancer learning a new pose, or a gymnast learning a fast and complex manoeuvre, acquiring new motor skills for humans typically involves guidance from another skilled human in the form of demonstrations. Acquiring skills from these demonstrations typically boils down to interpreting the individual features of the demonstration motion – for example, the relative positions of the limbs in a dance pose – and subsequently attempting to recreate the same features via repeated trial and error. Imitation learning (IL) is an algorithmic interpretation of this simple strategy of learning to imitate by matching the features of one's own motion with the features of the expert's demonstrations.

Such a problem can be solved by various means, with techniques like *behavioural cloning (BC)*, *inverse reinforcement learning (IRL)*, and their variants being popular choices. The imitation learning problem can also be formulated in various subtly differing ways, leading to different constraints on the types of algorithms that solve the problem. One notably challenging version of the problem is *Imitation from Observation* (IfO) [1], where the expert trajectories are only comprised of state features and no information about the expert's actions is available to the imitator. This means that learning a policy is not as straightforward as capturing the distribution of the expert's state-action pairs. Instead, the imitator must also learn to capture the dynamics of its environment. From a

practical perspective, the IfO problem is quite relevant as obtaining action-rich data for real-world tasks – across several agent embodiments and at large scales – is rather challenging. In most tasks, the expert only has an implicit representation of the policy. Imagine how a dancer cannot realistically convey their low-level actions – like muscle activations or positional targets – in a dance routine. Further, collecting action-rich data via teleoperation is time-consuming, requires access to robot hardware, and often produces unnatural motions owing to the challenges of conveying complex commands via a generally restrictive user interface. Imitation from observation hence closely depicts the data-limited reality of applying IL in the real world. Unfortunately, because BC relies on the expert's actions, a large fraction of BC techniques (including state-of-the-art algorithms like Diffusion Policy [2]) are inapplicable to the problem of imitating from observation. Inverse reinforcement learning, on the other hand, can still be applied to such problems.

In this work, we mainly focus on observation-based inverse reinforcement learning (reward learning), where the agent learns by recovering a scalar reward signal from the demonstrations. This learnt reward signal, when maximised by updating the agent's policy, provides the agent with the "correct" motivation to imitate the expert. While reward learning in itself is a broad field of study, recent works that leverage generative adversarial techniques for this task have shown markedly good results [3; 4; 1]. The fundamental idea in adversarial imitation learning (AIL) is to simultaneously learn and optimise the return from the reward function implied in the expert demonstrations through an optimisation objective derived from *Generative Adversarial Networks* (GANs) [5]. This procedure considers the agent's policy as a generator and parallelly trains a discriminator to differentiate between the motions in the expert demonstration dataset and the motions produced by the agent's policy. The discriminator's prediction is used as a reward signal in reinforcement learning and the policy and the discriminator are updated iteratively until convergence. This procedure is called min-max optimisation as the discriminator aims to minimise its classification error, while the generator aims to maximise the discriminator's error (either by generating high-quality samples or by directly maximising its rewards as in AIL). Although methods like AMP [3], GAIL [4], and GAIfO [1] have achieved impressive results in a wide variety of imitation tasks, they are prone to challenges intrinsic to their theoretical formulation. Adversarial training is known to suffer from a myriad of challenges [6; 7; 8; 5]. Primarily, these techniques are prone to have unstable training dynamics and learn non-smooth probability densities, with the source of these challenges attributed to the simultaneous min-max optimisation in the adversarial objective.

This paper explores a new generative framework for reward learning that completely sidesteps the limitations of adversarial imitation learning techniques. We use energy-based generative models as the backbone of our reward learning framework to learn smooth and accurate representations of the data distribution. We propose to use the learnt energy functions as reward functions and present a new imitation learning algorithm called *Noise-conditioned Energy-based Annealed Rewards (NEAR)* that has better, more predictable learning dynamics, and produces motions comparable to state-of-the-art adversarial imitation learning methods like AMP. Furthermore, to better ground the advantages of our work, we also closely analyse the challenges of using adversarial objectives in the context of imitation learning. We discuss previously under-examined failure modes in AIL and empirically verify our arguments on the Adversarial Motion Priors [3] algorithm. The primary contributions of this paper are

- Propose Noise-conditioned Energy-based Annealed Rewards (NEAR), a new state-only imitation learning algorithm based on energy-based generative modelling.
- Present theoretical arguments and empirical analysis to highlight the optimisation challenges of adversarial imitation learning algorithms and discuss some additional limitations of using adversarially learnt reward functions with reinforcement learning.

Before diving into our proposed framework (Sections 4 and 5), we first briefly discuss the challenges of adversarial IL in Section 2 and energy-based generative modelling in Section 3.

2 Background: Adversarial Imitation Learning

This paper refers to adversarial imitation learning as the family of techniques that use GAN-like objectives in their optimisation procedure. The current section briefly describes the details of this framework and discusses some prominent failure modes that lead to poor learning dynamics and instability.

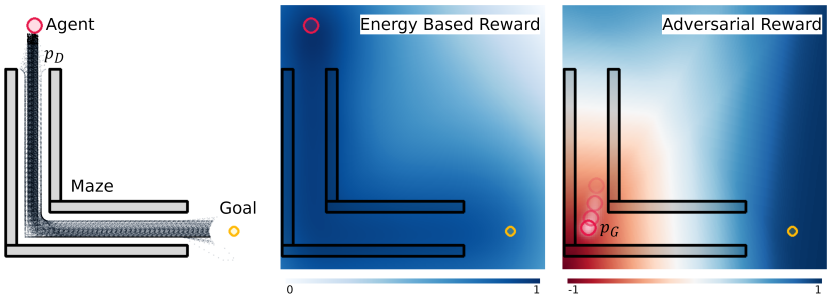

Figure 1: A comparison of reward functions (probability density approximations) learnt in a 2D target-reaching imitation task. Adversarial IL techniques learn a classifier between p_D and p_G which is non-smooth and prone to instability because of the changing nature of p_G (non-stationary). In contrast, energy-based models learn smooth and accurate representations of p_D and do not depend on p_G (stationary.)

GANs employ a simultaneous min-max optimisation procedure that aims to minimise the distance between two distributions, one being the distribution of data samples p_D (induced by a dataset $\mathcal{M}$) and the other being the learnt distribution p_G. To do this, GANs simultaneously learn parameterised functions called the *generator* and the *discriminator*. The generator typically produces new data samples x as a function of a white noise vector z [1], while the discriminator returns a scalar value representing the probability that x was derived from p_D. Overall the discriminator aims to correctly label both samples from p_D and p_G and the generator aims to return samples that the discriminator labels poorly [5]. In the context of imitation learning, $\mathcal{M}$ is a dataset of expert motion trajectories and p_D is a distribution of the features of these trajectories. The generator, however, does not generate these samples directly. Instead, the generator is a closed-loop policy, that returns an action as a function of the current state, that when applied to the environment, leads to features that resemble those in p_D. As a result, the input to the discriminator is *not* a direct output of the generator and there exists a functional disconnection (Appendix A.1) between the two. Owing to this, the generator in AIL algorithms is learnt via reinforcement learning [3; 4]. The complete procedure and the AIL optimisation problem are formalised below.

Given an expert motion dataset $\mathcal{M}$ containing i.i.d. data samples $x \equiv (s, s') \in X$ implying a distribution p_D, where X is the space of state transitions [2], adversarial IL methods aim to learn a differentiable generator (policy) $\pi_{\theta_G}(s) : S \to A$ where S is the state space, A is the action space, and $s \in S$ is a sample drawn from the occupancy measure of the policy ρ_π. Similarly to a standard GAN, the idea here is to learn a differentiable discriminator $D_{\theta_D}(x) : X \to [0, 1]$ that returns a scalar value representing the probability that the sample x was derived from p_D. However, now there exists an additional function $W(\pi_{\theta_G}(s)) : A \to X$ that maps the output of the policy to the discriminator's input space. The discriminator is learnt assuming that $W(\pi_{\theta_G}(s))$ is an i.i.d. sample in X and the generator is learnt via policy gradient methods by using $\log D_{\theta_D}(W(\pi_{\theta_G}(s)))$ as a reward function [3; 1].

$$\min_{\theta_D} \mathbb{E}_{x\sim p_D}[\log D_{\theta_D}(x)] + \mathbb{E}_{s\sim\rho_{\pi_{\theta_G}}}[\log(1 - D_{\theta_D}(W(\pi_{\theta_G}(s))))] \tag{1}$$

$$\max_{\theta_G} J \text{ where } \nabla_{\theta_G} J(\pi_{\theta_G}) = \mathbb{E}_{\pi_{\theta_G}}[Q^{\pi_{\theta_G}}(s,a)\nabla_{\theta_G}\log\pi_{\theta_G}(s,a)]$$

$$\text{where } Q^{\pi_{\theta_G}}(s,a) = \mathbb{E}_{\pi_{\theta_G}}[\log D_{\theta_D}(W(\pi_{\theta_G}(s)))]$$

Here J is the performance measure being maximised as per the policy gradient theorem [9]. The following section presents theoretical analyses that substantiate the claim that AIL algorithms suffer from similar optimisation challenges as standard GANs. A closer empirical verification of each argument can be found in Appendix A.3. In this paper, we argue that the following challenges are the most pertinent.

[1] z is a sample drawn from a known prior p_Z which, in most cases, this is just a noise distribution.

[2] In this paper we define all expressions for the partially observable case, however, the same results also apply to the fully observable cases.

- The generator suffers from poor training dynamics, especially at the initial stages of training. Additionally, training is quite unstable and the quality of the generated motions fluctuates drastically.
- In the context of reinforcement learning, the rewards learnt via adversarial techniques are non-smooth and do not always provide an unambiguous signal for improvement. Here, smoothness refers to the ability of a reward function to convey informative gradients in the sample space. It is important to note that the gradient of the reward function is not a direct part of the policy update in policy gradient methods, however, a smooth reward function is necessary for "sensible" policy updates.

2.0.1 Consequences of A Perfect Discriminator: Non-Smoothness, Non-Stationarity & Erratic Predictions

The iteratively changing nature of their training procedure and the formulation of the discriminator as a classifier are the root causes of the optimisation challenges in AIL algorithms. Throughout the training, the policy is updated to bring p_G closer to p_D, meaning that the support of p_G – the manifold of the samples generated by the policy – keeps changing. Simultaneously, the discriminator's decision boundary is also constantly changing to distinguish between the samples in $\text{supp}(p_D)$ and $\text{supp}(p_G)$. [8] present seminal theoretical work on the dynamics of the GAN optimisation procedure. Their work introduces the perfect discriminator theorems that state that if p_D and p_G have disjoint supports or have supports that lie in low-dimensional manifolds (lower than the dimension of the sample space X), then there exists an optimal discriminator $D^* : X \rightarrow [0, 1]$ that has accuracy 1 and $\nabla_x D^*(x) = 0 \forall x \in \text{supp}(p_D) \cup \text{supp}(p_G)$ (Theorems 2.1 and 2.2 in [8]). They also prove that under these conditions p_G is non-continuous in X and it is increasingly unlikely that $\text{supp}(p_D)$ and $\text{supp}(p_G)$ perfectly align (Lemmas 2 and 3 in [8]).

This means that at the initial stages of training, the discriminator very quickly learns to perfectly distinguish between the samples in the expert dataset $\mathcal{M}$ and those in p_G, assigning a prediction of 0 to any sample in the agent's trajectory. When used as a reward function, $\log D_{\theta_D}(W(\pi_{\theta_G}(s))) = \log 0$, instantly leads to arbitrary policy updates. Even when using a modified reward formulation, say $D()$ instead of $\log D()$, the agent would receive a nearly constant reward, say c. Under such a constant reward function, the gradient of the performance measure quickly goes down to zero – since the expectation of the gradient of the log probability of a parameterised distribution (Fisher score) is zero at the parameter [10]. We verify these claims by replicating the experiments from [8] on adversarial motion priors (AMP) (Appendix A.3.1).

$$Q^{\pi_{\theta_G}}(s, a) = \mathbb{E}_{\pi_{\theta_G}}[\log D_{\theta_D}(W(\pi_{\theta_G}(s)))] = \mathbb{E}_{\pi_{\theta_G}}[c] = c$$
$$\nabla_{\theta_G} J(\pi_{\theta_G}) = c \cdot \mathbb{E}_{\pi_{\theta_G}}[\nabla_{\theta_G} \log \pi_{\theta_G}(s, a)] = c \cdot 0 = 0$$

Furthermore, even without a perfect discriminator, the iteratively changing nature of adversarial learning leads to high-variance discriminator predictions and causes performance instability. At any point in training, the discriminator is trained to discriminate p_D from p_G – and *not* p_D from all that is not p_D – meaning that it is quite accurate on samples in $\text{supp}(p_D)$ and $\text{supp}(p_G)$ but is often arbitrarily defined in other regions of the sample space [8]. The changing nature of $\text{supp}(p_G)$ means that after a policy update, some of the samples being passed on to the discriminator could potentially have come from a region outside of these two supports. Since the discriminator is arbitrarily defined here, it is likely to return misleading predictions, leading to misleading policy updates. This variance is potentially further heightened by the stochastic nature of reinforcement learning techniques like Proximal Policy Optimisation (PPO) [11], meaning that the agent's exploration is typically met with poor rewards – we agree that the KL diverge constraint in PPO might somewhat reduce the negative impacts of this, however, penalising policy change with worse rewards is still non-ideal. The high variance hypothesis is also empirically verified with experiments on AMP (Appendix A.3.2).

Finally, we touch on the consequences of the non-smooth nature of the learnt reward function. Although smooth reward functions are not a mathematical necessity for policy gradient methods like PPO, reward smoothness is indeed an important criterion for faster convergence and sensible policy improvements. An ideal data-driven reward function is both smooth in the sample space as well as consistent throughout training (stationarity). Unfortunately, because D_{θ_D} is non-smooth in the whole

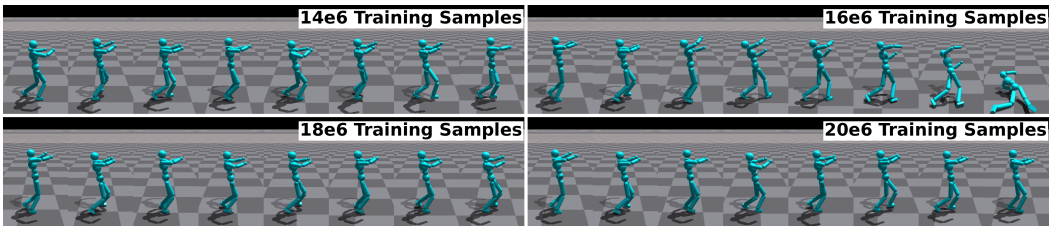

Figure 2: Policy degradation in the mummy style walking imitation task. With sufficient training, the policy does learn to complete the task, however, performance fluctuates substantially (with degradation seen at $16e6$ training samples).

space X and is arbitrarily defined in $(\text{supp}(p_D) \cup \text{supp}(p_G))^{\mathsf{c}}$ – parts of the sample space that are unexplored and not in the demonstration dataset $\mathcal{M}$ –, the rewards in this region are also non-smooth. As a result, an agent "stuck" in such a region would receive constant or arbitrarily changing rewards and hence the policy would receive uninformative updates. Moreover, it is also important for the reward function to be consistent in terms of the prediction assigned to a sample across different training iterations. However, because p_G and the discriminator's decision boundary are gradually changing, the learnt rewards are also non-stationary. If a sample is assigned a high reward at the initial stages and suddenly assigned a low reward at some point in training (or vice-versa), policy updates would also be rather misleading, regardless of the true desirability of that sample. We also carry out experiments to highlight the non-smoothness of the learnt reward function as well as its changes over training iterations (Appendix A.3.3).

To conclude this section, we point to Figures 1 and 2 that show a visual example of the non-smooth nature of the discriminator and qualitative results demonstrating policy instability. Having discussed several issues with adversarial IL, the next section introduces an alternative method of learning the expert's data distribution using which we subsequently propose a new imitation learning algorithm.

3 Score-Based Generative Models

Score-based generative models are a family of techniques that model the data distribution as a Boltzmann distribution and subsequently learn a parameterised function for the gradient of the log probability (score function) to generate samples [12]. Similarly to GANs, the aim here is to learn a probability distribution p_G that closely resembles the data distribution p_D, with $p_G \triangleq \frac{e^{-E(x)}}{Z}$ and $E(x)$ called the energy function of the distribution. Intuitively, the energy function is a measure of the closeness of a sample to p_D while the score is a vector pointing towards the steepest increase in the likelihood of p_D.

Because it is rather challenging to directly learn p_G, score-based models approximate the score function ($\nabla_x \log p_G(x)$) and use this to generate new samples with techniques like Markov Chain Monte Carlo sampling [13; 14]. Given data samples drawn from p_D, the score function is learnt by first perturbing the data samples by artificially adding noise and then learning a function to denoise a perturbed data sample by returning the vector that points towards the original sample – the denoising vector being the score of that perturbed sample. Learning the score function implicitly also learns the energy function as $\nabla_x \log p_G(x) = \nabla_x \log \frac{e^{-E(x)}}{Z} = -\nabla_x E(x)$. In this paper, we propose to explicitly learn the energy function to then use the energy of a sample to guide reinforcement learning. To do so, we make modifications to a score-based framework called Noise Conditioned Score Networks (NCSN) [15; 16].

3.1 Noise-Conditioned Score Networks (NCSN)

A score-based generative model requires three main components: samples drawn from a data distribution, a perturbation technique, and a way to learn the denoising process via a parameterised score function. Given i.i.d. data samples $\{x \sim p_D \in \mathbb{R}^D\}$, Noise Conditioned Score Networks [15] formulates a perturbation process that adds Gaussian noise $\mathcal{N}(x, \sigma)$ to each sample x, where σ is the standard deviation representing a diagonal covariance matrix and is sampled uniformly from a geometric sequence $\{\sigma_1, \sigma_2, ..., \sigma_L\}$. Following this perturbation process, we obtain a conditional

distribution $q_\sigma(x'|x) = \mathcal{N}(x'|x, \sigma I)$ from which a marginal distribution $q_\sigma(x')$ can be obtained as $\int q_\sigma(x'|x)p_D(x)dx$. Given this perturbed marginal distribution, NCSN attempts to learn a score function $s_\theta(x, \sigma) : \mathbb{R}^D \to \mathbb{R}^D$. The idea is to learn a conditional function to jointly estimate the scores of all perturbed data distributions, i.e., $\forall\sigma \in \{\sigma_i\}_{i=1}^L : s_\theta(x', \sigma) \approx \nabla_x \log q_\sigma(x')$. The score network is learnt via denoising score matching (DSM) [17] on samples drawn from the conditional distribution $q_\sigma(x'|x)$ – we leverage the fact that $\text{DSM}(q_\sigma(x')) = \text{DSM}(q_\sigma(x'|x)) + \text{const.}$ [12; 17]. The final DSM loss is averaged over the various σ values assigned to data samples in the training batch.

$$
\begin{aligned}
\text{DSM Loss} &= \mathbb{E}_{q_\sigma(x')}[\frac{1}{2}\|\nabla_x \log q_\sigma(x') - s_\theta(x', \sigma)\|] \\
&= \mathbb{E}_{q_\sigma(x'|x)}[\frac{1}{2}\|\nabla_x \log q_\sigma(x'|x) - s_\theta(x', \sigma)\|] + \text{const.} \\
&= \frac{1}{2}\mathbb{E}_{p_D}\mathbb{E}_{x'\sim\mathcal{N}(x,\sigma)}[\|\nabla_x \log q_\sigma(x'|x) - s_\theta(x', \sigma)\|] \quad \text{(drop const.)} \\
&= \frac{1}{2}\mathbb{E}_{p_D}\mathbb{E}_{x'\sim\mathcal{N}(x,\sigma)}[\|\frac{x' - x}{\sigma^2} - s_\theta(x', \sigma)\|] \qquad (2)
\end{aligned}
$$

By perturbing individual data samples with noise sampled from a Gaussian distribution with the mean as the data sample, NCSN essentially creates a perturbed distribution that is a smooth, dilated version of p_D (Figure 1). Gradually changing the standard deviation σ leads to a gradual change in the level of dilation. The use of this perturbation strategy has a few notable benefits.

- It is possible that p_D is supported in some low-dimensional manifold in the high-dimensional space $\mathbb{R}^D$. This was highlighted by previous experiments on the perfect discriminator and we saw that it is rather challenging to learn a continuous function over such a space as gradients are typically undefined in the ambient space. Adding a very small amount of Gaussian noise ensures that p_D is supported in the whole space, allowing the computation of gradients from which a score function can then be learnt.
- Even if p_D is supported in the whole space, it is possible that some regions are especially data-sparse. Since the expectation in Equation (2) is approximated by averaging over data samples, the learnt score function in these data-sparse regions could be misleading. The addition of a large amount of Gaussian noise can fill such data-sparse regions in p_D, leading to better score function approximation.
- From the point of view of reinforcement learning, it is better to learn dilated versions of p_D as opposed to learning a transformation from a Gaussian distribution to p_D (as in Diffusion [18; 19]). With reinforcement learning, the focus is not to generate new data samples but to use the energy function of the learnt distribution to guide an agent towards the true data distribution. As the agent always starts an episode from a starting state that is close to one of the expert's states, it is likely to also be in one of these perturbed distributions at the start of the episode (and quite unlikely to be in a zero-mean identity covariance Gaussian distribution, the sampling distribution of Diffusion). NCSN is hence more likely to provide meaningful guidance to the agent.
- Finally, NCSN has the added advantage of providing multiple, gradually dilated learnt distributions, using which an agent could be gradually motivated to converge to p_D.

4 Noise-conditioned Energy-based Annealed Rewards (NEAR)

The perturbed conditional distribution $q(x'|x)$ is formulated as a Boltzmann distribution such that $q(x'|x) = \frac{e^{\texttt{DIST}(x', x)}}{Z}$ where $\texttt{DIST}()$ is a function that defines some distance measure between a sample and its perturbed form. In NCSN, $\texttt{DIST}_\sigma()$ is the energy function of a Gaussian distribution with σ standard deviation and can be thought of as a dilated representation of the energy landscape of the expert data distribution p_D. Our approach leverages the fact that for a sample $x \in X$, $\texttt{DIST}_\sigma(x)$ is essentially just a scalar-valued measure of the closeness of x to p_D, meaning that it can be used as a reward signal to guide a policy to generate motions that gradually resemble those in p_D. Unlike

AMP, we propose to train NCSN *before* training the reinforcement learning policy and subsequently use the learnt energy function as a reward function. There are several significant advantages to this:

- Since the energy function is learnt via score matching on samples arbitrarily far away from the p_D, $\texttt{DIST}_\sigma()$ is both well-defined and continuous in the relevant parts of the sample space X. Continuity in the whole space indeed requires an infinitely large σ, however, realistically a σ that is sufficiently large to cover the worst-case policy would guarantee continuity in the relevant parts of X.
- Since $\texttt{DIST}_\sigma()$ is a dilated version of the energy function of p_D, it is also not prone to being constant valued. It will indeed be arbitrarily defined for samples that are very far from p_D (because those samples were never passed on to the network), however again, a sufficiently large σ would mean that it is non-constant in the parts of X where the policy-generated samples are realistically expected to lie.
- Finally, $\texttt{DIST}_\sigma()$ is learnt using perturbed samples from p_D and hence does not rely on the policy-generated samples. This means that it is disconnected from the policy and is not prone to issues of high variance that come with simultaneous training. Further, we propose to learn $\texttt{DIST}_\sigma()$ in a one-shot manner, eliminating any concerns of non-stationarity.

The following sections discuss the procedure to learn these energy functions and other algorithmic details of our approach (Algorithm 1).

4.1 Learning Energy Functions

This paper modifies NCSN [15] to learn energy functions instead of score functions. Given D-dimensional i.i.d. data samples $\{x \sim p_D \in \mathbb{R}^D\}$ where p_D is the distribution of state-transition features in the expert's trajectories, NEAR learns a parameterised energy function $e_\theta(x', \sigma) : \mathbb{R}^D \rightarrow \mathbb{R}$ that approximates the energy of samples x' in a perturbed data distribution obtained by the local addition of Gaussian noise $\mathcal{N}(x, \sigma)$ to each sample x. The idea here is to jointly estimate the energy functions of several perturbed distributions, i.e., $\forall \sigma \in \{\sigma_i\}_{i=1}^L : e_\theta(x', \sigma) \approx \texttt{DIST}_\sigma(x')$. The sample's score is computed by taking the gradient of the predicted energy w.r.t the perturbed sample, $s(x', \sigma) = \nabla_{x'} e_\theta(x', \sigma)$. The energy network is learnt via denoising score matching using this computed score and the final DSM loss in a training batch is computed as an average over the various σ values assigned to data samples in the training batch.

$$
\begin{aligned}
l_{\text{DSM}}(\sigma) &= \frac{1}{2}\mathbb{E}_{p_D}\mathbb{E}_{x' \sim \mathcal{N}(x,\sigma)}[\| \frac{x' - x}{\sigma^2} - \nabla_{x'} e_\theta(x', \sigma)\|] \\
\mathcal{L}_{\text{DSM}}(\{\sigma_i\}_{i=i}^L) &\triangleq \frac{1}{L}\sum_{i=1}^{L} l_{\text{DSM}}(\sigma_i)
\end{aligned} \tag{3}
$$

We modify the definition of the perturbed conditional distribution $q(x'|x)$ by flipping the sign of the energy function such that higher energies indicate closeness to p_D. This is done to simplify the downstream reinforcement learning such that the predicted energy can be maximised directly. Following the improvements introduced in [16], we define $e_\theta(x', \sigma) = \frac{e_\theta(x')}{\sigma}$ where $e_\theta(x')$ is an unconditional energy network [3]. This allows us to learn the energy function of a large number of noise scales with a very small sized dataset.

The appropriate selection of the noise scale ($\{\sigma_i\}_{i=i}^L$) is highly important for the success of this framework. σ_L must be small enough that the perturbed distribution $q_{\sigma_L}()$ is nearly identical to p_D. This ensures that the policy aims to truly generate samples that resemble those in p_D. In contrast, σ_1 must be sufficiently large such that $e_\theta(x', \sigma_1)$ is well-defined, continuous, and non-zero for any sample that is generated by the worst-possible policy. This ensures that the agent always receives an informative signal for improvement. Assuming that policy degradation is unlikely, σ_1 must be such that $\text{supp}(q_{\sigma_1}())$ effectively contains the support of the distribution induced by a randomly initialised policy network. In practice, these are dataset-dependent hyperparameters.

[3] The score is the gradient of the energy function and the norm of the score scales inversely with σ. The score can hence be approximated by rescaling the energy with $\frac{1}{\sigma}$ [16].

Algorithm 1 Noise-conditioned Energy-based Annealed Rewards

Require: Dataset of reference motions, $\mathcal{M} \equiv \{(s, s')\}$
Require: NCSN noise scale $\{\sigma_i\}_{i=i}^{L}$ (σ_1 is highest std. and σ_L is lowest std.)
Require: Threshold percentage to change noise level T (say 0.05)
Ensure: Energy network e_θ, policy π_{θ_G}, and value fn. V are initialised
Ensure: Replay buffer $\mathcal{B} \leftarrow \emptyset$ and annealing buffer $\mathcal{A} \leftarrow \emptyset$ are initialised
Ensure: Current noise level $\sigma_k = \sigma_1$, mean initial return R'_{σ_k}, number of iterations after noise level change $N_{iters} = 0$ are initialised
Ensure: Mean return of the last β training iterations with the current noise level R_{σ_k} is initialised

procedure NCSN($\mathcal{M}$, noise scale)
  for update step = $1, \ldots, N_{iters}$ **do**
    $b^{\mathcal{M}} \leftarrow$ sample a batch of transitions from $\mathcal{M}$
    $b^{sigma} \leftarrow$ sample nose levels σ_k for every sample in $b^{\mathcal{M}}$ uniformly from $\{\sigma_i\}_{i=i}^{L}$
    Update e_θ according to Equation (3) using pairing $b^{\mathcal{M}} : b^{sigma}$

procedure RL(learnt e_θ)
  Initialise starting nose level $\sigma_k = \sigma_1$
  while not done **do**
    for trajectory = $1, \ldots, m$ **do**
      $\tau_i \leftarrow \{[s, a, s', r' = \texttt{rew-tf}(e_\theta(s, s', \sigma_k))]^{\text{till horizon}}\}_{\pi_{\theta_G}}$ ▷ `rew-tf` - Section 5.1
      Store τ_i in $\mathcal{B}$
      Store $\{(s, s')^{\text{till horizon}}\}$ in $\mathcal{A}$
    $\sigma_k \leftarrow \texttt{anneal}(\mathcal{A})$ ▷ `anneal` - Section 4.2
    Reset $\mathcal{A} \leftarrow \emptyset$
    Update V and π_{θ_G} by sampling trajectories from $\mathcal{B}$

procedure ANNEAL(anneal buffer $\mathcal{A} = \{(s, s')\}$)
  N_{iters} += 1
  if $N_{iters} < \beta$ **then**
    Update R'_{σ_k} with $\texttt{mean}(e_\theta(\mathcal{A}, \sigma_k))$
  Update R_{σ_k} with $\texttt{mean}(e_\theta(\mathcal{A}, \sigma_k))$
  if $N_{iters} > \beta$ **then**
    if $R_{\sigma_k} \geq (1 + T) \cdot R'_{\sigma_k}$ **then**
      Increase σ_k ▷ Do not change if at σ_L
      Reset R_{σ_k} and R'_{σ_k}. $N_{iters} = 0$
    else if $R_{\sigma_k} < (1 - T) \cdot R'_{\sigma_k}$ **then**
      Decrease σ_k ▷ Do not change if at σ_1
      Reset R_{σ_k} and R'_{σ_k}. $N_{iters} = 0$

4.2 Annealing

The trained energy network $e_\theta(x', \sigma)$ can directly be used as a reward function to train a policy network π_{θ_G} (say initialised at θ_{G0}) using some fixed noise level σ_k. But how do we decide on an appropriate σ_k? Assuming that during training the noise scale was set appropriately, $q_{\sigma_L}()$ is nearly identical to p_D but $\text{supp}(q_{\sigma_L}())$ has a low intersection with $\text{supp}(\pi_{\theta_{G0}})$ [4]. On the other hand $q_{\sigma_1}()$ is an extremely dilated version of p_D but $\text{supp}(q_{\sigma_1}())$ is likely to have a high intersection with $\text{supp}(\pi_{\theta_{G0}})$. This means that any chosen noise-level σ_k offers a tradeoff between sample quality and $e_\theta(x', \sigma_k)$ being continuous and well-defined in the manifold of the samples generated by the current policy.

We propose an annealing framework inspired by annealed Langevin dynamics and its predecessors [15; 20; 21] to ensure that the learnt reward function is always well-defined and continuous while also gradually changing to motivate the policy to get closer to p_D. Instead of focusing on sample

[4] As a shorthand, we abbreviate the support of the distribution of state transitions induced by rolling out a policy as $\text{supp}(\pi_{\theta_G})$.

generation, annealing in the context of reinforcement learning focuses on making gradual changes to the agent's reward function. Our annealing framework hence depends on the agent's progress from an imitation perspective. Training is initialised with the energy function of the lowest noise level. Then, at every new noise level, the agent tracks the average return of the first few policy updates. The noise level is increased if the average return of the last few policy updates is higher than some percentage of the initial return. We note that changing the reward function introduces non-stationarity in the reinforcement learning problem, meaning that the learnt policy is susceptible to degradation. To account for this, our framework also lowers the noise level if the return drops below some percentage of the initial return. This means that if the policy gets worse, the noise level decreases, thereby increasing the intersection between $\text{supp}(q_\sigma())$ and $\text{supp}(\pi_{\theta_G})$ and ensuring that the degraded policy still has an informative reward signal for improvement.

5 Experiments

The following subsection discusses the implementation details of NEAR, the experimental setup used in this paper, and other information like performance metrics and comparisons. Additional details of the experimental implementation can be found in Appendix C and a detailed reproduction guide is available in the documentation of our opensource codebase (github.com/anishhdiwan/near). Section 5.2 presents our results and a discussion on some notable trends which is followed by ablation experiments in Section 5.3.

5.1 Experimental Setup

We evaluate NEAR (Algorithm 1) on complex, physics-dependent, contact-rich humanoid motions such as stylised walking, running, and martial arts. The chosen task set demands an understanding of physical quantities such as gravity and the mass/moments of inertia of the character and contains a variety of fast, period, and high-acceleration motions. The expert's motions are obtained from the CMU and SFU motion capture datasets and contain trajectories of several motions. For each motion, a dataset of state transitions $\mathcal{M} \equiv \{(s, s')\}$ is created to learn an imitation policy.

$$\tilde{r}(s, a, s', g) = w^{task} r^{task}(s, a, g) + w^{energy} e_\theta(s, s') \tag{4}$$

To understand the impact of motion data availability on the algorithm, we also train NEAR in a single-clip setting – using a single expert motion for training – on challenging motions like mummy-style walking and spin-kick. Further, to understand the composability of the learnt rewards, we train NEAR with both environment-supplied rewards (such as a target reaching reward) and energy-based rewards learnt from different motion styles (to perform hybrid stylised motions). To incorporate the environment-supplied task reward $r^{task}(s, a, g) \in [0, 1]$, we use the same strategy from [3] and formulate learning as a goal-conditioned reinforcement learning problem, where the policy is now conditioned on a goal g and maximises a reward $\tilde{r}(s, a, s', g)$ (Equation (4)). Details of the tasks and goals can be found in Appendix B. We also apply an additional reward transformation of $\tanh\left(\frac{\tilde{r}-r'}{10}\right)$ where r' is the mean horizon-normalised return received by the agent in the last $k = 3$ policy iterations, $r' = \texttt{mean}(\{\frac{\tilde{R}_{t-i}}{horizon}\}_{i=1}^{k})$. This bounds the unnormalised energy reward to a fixed interval so that changes between the noise levels σ are smoother. Additionally, it grounds the agent's current progress in relation to its average progress in the last few iterations. The policy is trained using Proximal Policy Optimisation [11] and we use the following quantitative metrics to measure the performance of our algorithm.

Average Dynamic Time Warping Pose Error: This is the mean dynamic time warping (DTW) error [22] between trajectories of the agent's and the expert's poses averaged across all expert motions in the dataset. The DTW error is computed using $\|\hat{x}_t - x_t\|_2$ as the cost function, where $\hat{x}_t$ and x_t are the Cartesian positions of the reference character and agent's bodies at time step t. To ensure that the pose error is only in terms of the character's local pose and not its global position in the world, we transform each Cartesian position to be relative to the character's root body position at that timestep ($\hat{x}_t \leftarrow \hat{x}_t - \hat{x}_t^{root}$ and $x_t \leftarrow x_t - x_t^{root}$).

Spectral Arc Length: Spectral Arc Length (SAL) [23; 24; 25] is a measure of the smoothness of a trajectory and is an interesting metric to determine the policy's ability to perform periodic motions

in a controlled manner. SAL relies on the assumption that smoother motions are comprised of fewer and low-valued frequency domain components while jerkier motions have a more complex frequency domain signature. SAL is computed by adding up the lengths of discrete segments (arcs) of the normalised frequency-domain map of a motion. In our experiments, we use SPARC [23], a more robust version of the spectral arc length that is invariant to the temporal scaling of the motion. Note that in this case, we do not transform the positions to the agent's local coordinate system.

Root Body Position Derivatives: Finally, we also compute the average velocity and jerk of the character's root body (hip) and match this with those of the expert motions. The learnt rewards favour velocity similarity with the expert's motions, however, a similarity in jerk is not directly maximised.

5.2 Results

We compare Noise-conditioned Energy-based Annealed Rewards (NEAR) with Adversarial Motion Priors (AMP) and in both cases only use the learnt rewards to train the policy. Each algorithm-task combination is trained 5 times independently and the mean performance metrics across 20 episodes of each run are compared. Both algorithms are trained until some pre-defined maximum training samples.

Figure 3 shows snapshots of the policies trained using NEAR. We find that NEAR achieves very close imitation performance with the expert's trajectory and learns policies that are visually smoother and more natural (videos here). For the quantitative metrics, we use the average performance at the end of training for a fair comparison and find that both NEAR and AMP are roughly similar across all metrics (Table 1). In most experiments, NEAR is closer to the expert in terms of the spectral arc length, root body velocity, and jerk (Figures 4 and 5) while AMP has a better pose error. NEAR also outperforms AMP in stylised goal-conditioned tasks, producing motions that both imitate the expert's style while simultaneously achieving the desired global goal (Table 2 and Figure 3 bottom). From the experiments on spatially composed learnt rewards, we find that NEAR can also learn hybrid policies such as waking while waving. Finally, we notice that NEAR performs poorly in single-clip imitation tasks, highlighting the challenges of accurately capturing the expert's data distribution in data-limited conditions. Conversely, AMP is less affected by data unavailability since the discriminator in AMP is simply a classifier and does not explicitly capture the expert's distribution. An extended set of results is provided in Appendix D.

In the experiments on NEAR, it was noticed that the pose error seemingly grows at the early stages of training. This is counterintuitive as the energy return and visual inspection show that the policy-generated motions indeed get better over time. The reason behind this could be the large dilation of the initial perturbed distributions $q_{\sigma_k}()$ compared to p_D. Even though the policy is maximising the energy function of these highly dilated perturbed distributions, the policy itself might be so far from p_D that the motions in $\text{supp}(\pi_{\theta_G})$ do not exactly resemble those in the expert dataset. However, annealing over time does lead to a slow shift in $\text{supp}(\pi_{\theta_G})$ towards p_D. Hence, we believe that this trend does not necessarily indicate a drawback of NEAR but rather shows that a larger importance is needed for other metrics during the early stages of training.

The performance of NEAR was also seen to become unpredictable as training progressed beyond a certain point (typically after 40e6 training steps). We attribute this to issues with annealing at higher noise levels. We hypothesise that NEAR learns quite accurate policies before exhausting the complete noise scale. Meaning that the policy is the closest to the expert's motions midway through training. Training the policy beyond this point while still annealing the noise level leads to the introduction of drastic non-stationarity in the problem. This could be because of the geometric nature of the noise scale which causes the perturbed distributions to change drastically at higher levels. While at the early stages, the supports of these perturbed distributions are large enough to still encompass the manifold of the policy-generated motions after a noise level change, the supports of the later perturbed distributions might only partially contain this manifold. This means that a change in noise level suddenly causes the energy function to be ill-defined on a portion of the manifold of policy-generated motions, leading to poor rewards for these transitions. This can be verified with the energy return plot, where the return often drops at noise level changes indicating that the changed noise level is suddenly low-rewarding (Figure 6). This is further corroborated by the ablations that follow in the next section, where it can be seen that the configurations without annealing are generally more predictable at the later stages of training. Degradation at higher noise levels highlights the sensitivity of NEAR to the noise scale and is a limitation of this framework. Improvements can perhaps be

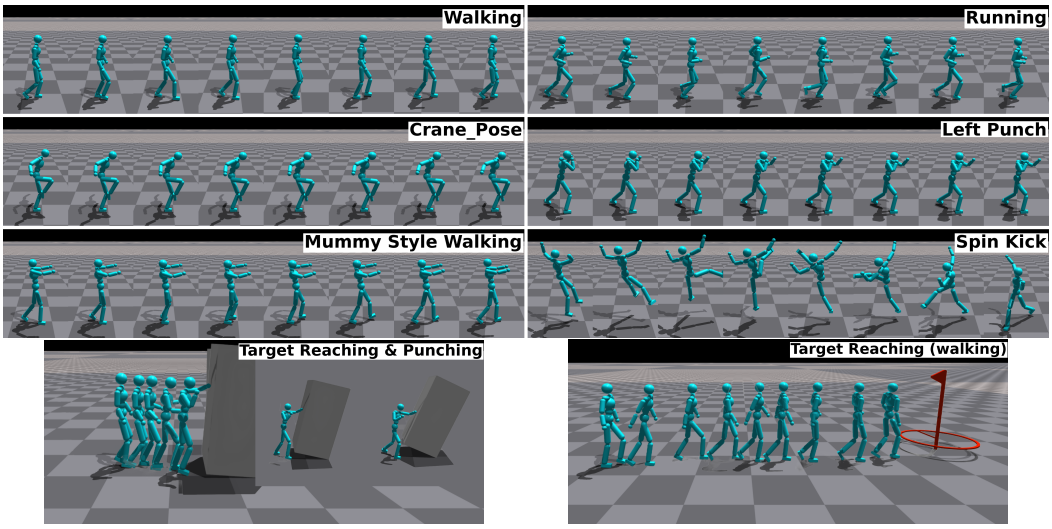

Figure 3: Snapshots of the policies trained with NEAR. Mummy-style walking and spin-kick are single-clip imitation tasks. The bottom row shows goal-conditioned RL policies that also optimise an environment-provided task reward.

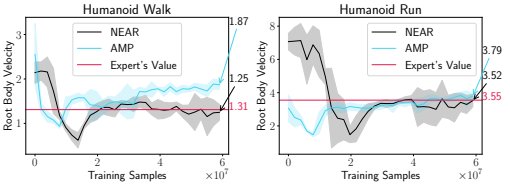

Figure 4: The character's root body velocity throughout training.

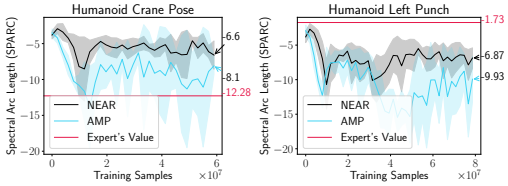

Figure 5: The character's spectral arc length throughout training.

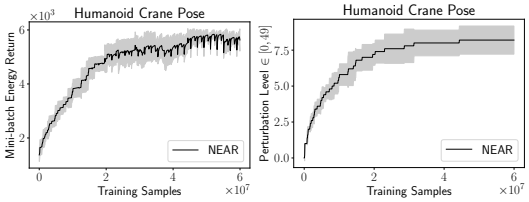

Figure 6: Annealing at higher noise levels causes a drop in the energy reward's return, indicating that the new reward function could be poorly defined on a portion of the manifold of policy-generated motions.

made by experimenting with a linear noise scale, even more noise levels, or a more dynamic form of annealing. However, we leave these improvements for future work.

5.3 Ablations

We also conduct ablation experiments that help identify the crucial components of NEAR. The main focus of these experiments is to understand the contributions of annealing and the effects of using an environment-provided task reward (no goal-conditioning). For walking and running, the task reward favoured forward motion and episode length while for the crane pose task it only favoured episode length. Further, we also ablate parts of the NCSN algorithm to understand the benefits of the strategies discussed in [16]. Specifically, instead of defining the energy function using an unconditional energy network, we directly train a conditional energy network $e_\theta(x', \sigma)$. We also

Table 1: A comparison of the avg. pose error (lower is better) and spectral arc length (closer to expert is better) at the end of training.

Task	Num. Motion Clips	Algorithm	Avg. Pose Error (m)	Spectral Arc Length
Walking	74 (249.38 sec.)	NEAR	**0.51**	**-7.52**
		AMP	**0.51**	-8.78
		Expert's Value		-5.4
Running	26 (32.55 sec.)	NEAR	**0.62**	**-7.24**
		AMP	0.65	-9.71
		Expert's Value		-3.79
Crane Pose	3 (24.62 sec.)	NEAR	0.94	-6.6
		AMP	**0.82**	**-8.1**
		Expert's Value		-12.28
Left Punch	19 (4.91 sec.)	NEAR	0.37	**-6.87**
		AMP	**0.32**	-9.93
		Expert's Value		-1.73
Mummy Walk	1 (5.40 sec.)	NEAR	0.66	**-4.72**
		AMP	**0.41**	-13.84
		Expert's Value		-4.71
Spin Kick	1 (1.15 sec.)	NEAR	0.78	-5.59
		AMP	**0.58**	**-3.16**
		Expert's Value		-3.39

Table 2: A comparison of the avg. pose error (lower is better) and task return (higher is better) at the end of training with temporally and spatially composed reward functions.

Task	Algorithm	Avg. Pose Error (m)	Task Return
Target Reaching (walking)	NEAR	**0.94**	**2.75**
	AMP	1.09	2.23
Target Reaching (running)	NEAR	**1.18**	**1.74**
	AMP	1.77	-0.15
Target Reaching & Punching	NEAR		3.6
	AMP		**3.85**

Task	Algorithm	Avg. Walking Pose Error (m)	Avg. Waving Pose Error (m)
Walking & Waving	NEAR	1.33	0.86

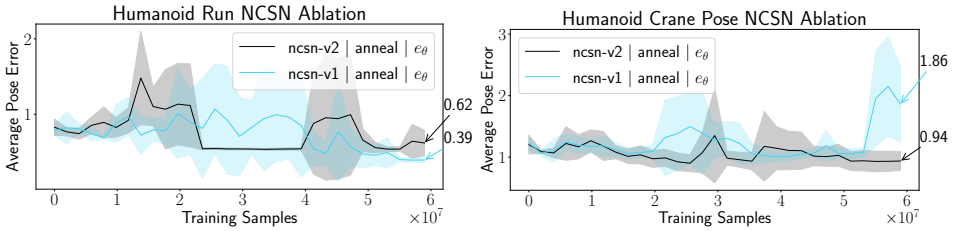

Figure 7: The average pose error of the ablated NCSN configuration throughout training. Notice that the ncsn-v1 configuration has a higher variance.

redefine $\sigma_1 = 10$ as per [15] and reduce the noise scale to only have 10 noise levels. We hypothesise that the modified NCSN configuration – which we call NCSN-v1 – would learn a poor energy function because of the challenges of learning a complex conditional function with limited data. Further, we hypothesise that this would poorly impact reinforcement learning because $\text{supp}(q_{\sigma_1}())$ would have a lower intersection with $\text{supp}(\pi_{\theta_{G0}})$ and the change between noise scales would be much more pronounced (thereby leading to poor annealing). Given these parameters of interest, we train five ablated configurations of NEAR (with 5 independent runs each and performance averaged over 20 episodes of each run) and show some of the key results in Table 3 and Figure 7 (full list of results in Appendix E).

Table 3: A comparison of ablated configurations of NEAR. Shaded rows compare the effect of annealing (boldface is better) while the non-shaded regions compare the addition of environment-provided task rewards (underlined is better).

Task	Ablated Configuration	Avg. Pose Error (m)	Spectral Arc Length
Walking	σ_5 & e_θ	0.49	<u>-7.1</u>
	σ_5 & $\tilde{r}$	<u>0.42</u>	-8.7
	anneal & e_θ	0.51	-7.52
	σ_5 & e_θ	**0.49**	**-7.1**
	Expert's Value		-5.4
Running	σ_5 & e_θ	0.62	<u>-6.69</u>
	σ_5 & $\tilde{r}$	<u>0.57</u>	-8.61
	anneal & e_θ	**0.62**	-7.24
	σ_5 & e_θ	**0.62**	**-6.69**
	Expert's Value		-3.79
Crane Pose	σ_5 & e_θ	1.38	<u>-6.03</u>
	σ_5 & $\tilde{r}$	<u>1.23</u>	-4.34
	anneal & e_θ	**0.94**	**-6.6**
	σ_5 & e_θ	1.38	-6.03
	Expert's Value		-12.28

First, these experiments provide conclusive evidence of the improvements obtained by the NCSN modifications from [16]. In all cases, the ablated configuration had highly unstable learning dynamics and often also had poor average performance. From visual inspection of the learnt policies, it was apparent that the ablated version's policies were rather non-smooth and often failed mid-episode. The policy was also prone to converging to locally optimal behaviours similar to the ones highlighted in [3], where the character would simply jitter forwards and backwards while not actually moving. This jittering might also be the reason for the unusually low spectral arc length and pose error in some tasks.

Secondly, it can be seen that the addition of the task reward leads to an improvement in the pose error. This is especially apparent in tasks like walking and running where the task reward is closely aligned with the imitation objective. The addition of the task reward also counteracted the unpredictability of NEAR at the later stages of training, which again corroborates the hypothesis that the source of this instability was the non-stationarity introduced by annealing. It must however be noted that the addition of the task reward does mean that the agent has a reduced closeness to specific characteristics of the expert's motion like spectral arc length, velocity, and jerk. Ultimately the closeness of the imitation under a combined reward still highly depends on the harmony between the two reward functions.

Finally, annealing does not have a significant impact on performance in walking and running, however, leads to an improvement in more complex, non-periodic cases like the crane-pose task. It is possible that for complex tasks, the expert distribution is more densely concentrated. In this case, a higher noise level for a complex task might be more informative than one for a simpler task for which the expert distribution is more spread out. The increased information available from annealing might hence be the reason for better results with annealing in complex tasks. We also notice that in configurations with an added task reward, annealing also does not cause any significant policy degradation. A reason for this could be the reduction in non-stationarity by the addition of a non-changing component to the reward function.

6 Limitations & Conclusions

While NEAR is capable of generating high-quality, life-like motions and also outperforms AMP in several tasks, it is still prone to some limitations. The annealing strategy discussed in Section 4.2 does lead to progressively improving rewards, however, effective annealing at high noise levels is still a challenge. Results from Section 5.2 highlight that annealing the energy function when the policy-generated motions are close to convergence leads to a drop in the received reward, thereby leading to policy degradation. While the fundamental idea behind annealing might not be flawed,

there is certainly scope for an improved mechanism for changing noise levels that does not cause the rewards to be ill-defined for the current policy.

State-only reward learning techniques like NEAR and AMP are also quite sensitive to the motion dataset. We find that the policy is prone to converging to locally optimal behaviour if the dataset contains bi-directional state transitions (such that (s, s') and (s', s) are equally likely to occur). We found greater success when the motion clips of repeating tasks were temporally cropped to only contain a few cycles of the motion. Another limitation of state-only imitation reward learning is obtaining adequate exploration. We found that actions such as jumping that require sudden and explosive actions took increasingly long to learn. The reason for this could be the need for having explored a very specific action at a very specific state – say a suddenly large positional target at the jumping instance. Finally, we find that the quality of the learnt policy varies depending on the initial state from which the rollouts are started. The policy is fairly robust in imitating the expert starting from a state in the expert's motions, however, it often fails if the initial state is out of the distribution. For example, punching from a "defensive stance" is quite robust but punching from a "standing position" fails easily. Again, we attribute these issues to a lack of exploration and hypothesise that longer training runs would iron out these inconsistencies. Unfortunately, because of the resource-contained nature of our research, we were not able to run very long training experiments. Further, it must be noted that these trends were true for both NEAR and AMP.

To conclude, this paper proposes an energy-based framework for imitation learning in partially observable conditions. Our framework builds on Noise-conditioned Score Networks [15] to explicitly learn a series of smooth energy functions from a dataset of expert demonstration motions. We propose to use these energy functions as reward functions to learn imitation policies via reinforcement learning. Further, we propose an annealing framework for reinforcement learning tasks to gradually change the learnt reward functions as the policy improves. Our proposed imitation learning algorithm called Noise-conditioned Energy-based Annealed Rewards (NEAR) outperforms state-of-the-art methods like Adversarial Motion Priors (AMP) in several quantitative metrics as well as qualitative evaluation across a series of complex contact-rich human imitation tasks. To ground the advantages of our work, a part of this paper also presents a series of theoretical arguments and empirical results that show the various optimisation-related challenges of adversarial imitation learning algorithms.

Acknowledgements

- The authors acknowledge the use of computational resources of the DelftBlue supercomputer, provided by Delft High Performance Computing Centre (https://www.tudelft.nl/dhpc).
- The data used in this project was obtained from mocap.cs.cmu.edu. The database was created with funding from NSF EIA-0196217.
- The data used in this project was obtained from mocap.cs.sfu.ca. The database was created with funding from NUS AcRF R-252-000-429-133 and SFU President's Research Start-up Grant.

References

[1] F. Torabi, G. Warnell, and P. Stone, "Generative adversarial imitation from observation," *arXiv preprint arXiv:1807.06158*, 2018.

[2] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *arXiv preprint arXiv:2303.04137*, 2023.

[3] X. B. Peng, Z. Ma, P. Abbeel, S. Levine, and A. Kanazawa, "Amp: Adversarial motion priors for stylized physics-based character control," *ACM Transactions on Graphics (ToG)*, vol. 40, no. 4, pp. 1–20, 2021.

[4] J. Ho and S. Ermon, "Generative adversarial imitation learning," *Advances in neural information processing systems*, vol. 29, 2016.

[5] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[6] D. Saxena and J. Cao, "Generative adversarial networks (gans) challenges, solutions, and future directions," *ACM Computing Surveys (CSUR)*, vol. 54, no. 3, pp. 1–42, 2021.

[7] N. Kodali, J. Abernethy, J. Hays, and Z. Kira, "On convergence and stability of gans," *arXiv preprint arXiv:1705.07215*, 2017.

[8] M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks," *arXiv preprint arXiv:1701.04862*, 2017.

[9] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in neural information processing systems*, vol. 12, 1999.

[10] L. Wasserman, *All of statistics: a concise course in statistical inference*. Springer Science & Business Media, 2013.

[11] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[12] Y. Song and D. P. Kingma, "How to train your energy-based models," *arXiv preprint arXiv:2101.03288*, 2021.

[13] G. Parisi, "Correlation functions and computer simulations," *Nuclear Physics B*, vol. 180, no. 3, pp. 378–384, 1981.

[14] U. Grenander and M. I. Miller, "Representations of knowledge in complex systems," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 56, no. 4, pp. 549–581, 1994.

[15] Y. Song and S. Ermon, "Generative modeling by estimating gradients of the data distribution," *Advances in neural information processing systems*, vol. 32, 2019.

[16] Y. Song and S. Ermon, "Improved techniques for training score-based generative models," *Advances in neural information processing systems*, vol. 33, pp. 12438–12448, 2020.

[17] P. Vincent, "A connection between score matching and denoising autoencoders," *Neural computation*, vol. 23, no. 7, pp. 1661–1674, 2011.

[18] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[19] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *International conference on machine learning*, pp. 2256–2265, PMLR, 2015.

[20] S. Kirkpatrick, C. D. Gelatt Jr, and M. P. Vecchi, "Optimization by simulated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983.

[21] R. M. Neal, "Annealed importance sampling," *Statistics and computing*, vol. 11, pp. 125–139, 2001.

[22] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE transactions on acoustics, speech, and signal processing*, vol. 26, no. 1, pp. 43–49, 1978.

[23] Y. Beck, T. Herman, M. Brozgol, N. Giladi, A. Mirelman, and J. M. Hausdorff, "Sparc: a new approach to quantifying gait smoothness in patients with parkinson's disease," *Journal of neuroengineering and rehabilitation*, vol. 15, pp. 1–9, 2018.

[24] S. Balasubramanian, A. Melendez-Calderon, and E. Burdet, "A robust and sensitive metric for quantifying movement smoothness," *IEEE transactions on biomedical engineering*, vol. 59, no. 8, pp. 2126–2136, 2011.

[25] S. Balasubramanian, A. Melendez-Calderon, A. Roby-Brami, and E. Burdet, "On the analysis of movement smoothness," *Journal of neuroengineering and rehabilitation*, vol. 12, pp. 1–11, 2015.

[26] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine learning*, vol. 3, pp. 9–44, 1988.

[27] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[28] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, JMLR Workshop and Conference Proceedings, 2010.

[29] J. Choquette, W. Gandhi, O. Giroux, N. Stam, and R. Krashinsky, "Nvidia a100 tensor core gpu: Performance and innovation," *IEEE Micro*, vol. 41, no. 2, pp. 29–35, 2021.

[30] Delft High Performance Computing Centre (DHPC), "DelftBlue Supercomputer (Phase 2)." https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase2, 2024.

[31] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, *et al.*, "Isaac gym: High performance gpu-based physics simulation for robot learning," *arXiv preprint arXiv:2108.10470*, 2021.

A Optimisation Challenges in Adversarial IL

A.1 Distinctions Between Adversarial IL and Standard GANs

As explained in Section 2, in AIL methods, the input to the discriminator is *not* a direct output of the generator, meaning that the generator does not directly generate samples in expert data space. Instead, the generator is a policy π_{θ_G} that returns actions. The input to the discriminator is typically obtained by passing these actions to another arbitrary function $W()$ that maps the generator's output to the discriminator's sample space X. The key problem is that $W()$ is potentially unknown and is not guaranteed to be differentiable. This does not pose a problem to the discriminator as updates to the discriminator do not rely on the gradients of the generator –it can be assumed that $W(\pi_{\theta_G}(s))$ is just another i.i.d. sample in X. However, if the gradients from the discriminator can not flow back to the generator, how then is the generator (policy) updated? The generator is learnt via policy gradient methods by using the discriminator's prediction as a reward function. The policy gradient theorem ensures that the gradients of the reward function (discriminator) are not a part of the policy update.

The adversarial optimisation procedure in AIL algorithms is hence, not exactly the same as it is in their conventional sample-generation-focused counterparts – the primary difference being the use of reinforcement learning and the disconnected gradient flow. Given this distinction, do adversarial IL algorithms still suffer from the same training limitations? Our theoretical analysis in Section 2.0.1 and the supplementary experimental analysis in Appendix A.3 show this is indeed true.

A.2 Perfect Discriminator Theorems Elaboration

To fully understand the points of failure in GAN training, let us first look at the mechanics of the discriminator's and the generator's objectives. Given samples drawn from a data distribution p_D, the idea of a GAN is to learn a generator that induces a distribution p_G. The generator aims to iteratively change p_G to bring it closer to p_D while the discriminator aims to update its weights to maximise the distance (typically, the Jensen Shannon Divergence) between the two. Changing p_G refers to the process of changing its support to contain samples that progressively resemble those in the support of p_D. At the end of training the goal is to have a high intersection between $\text{supp}(p_D)$ and $\text{supp}(p_G)$.

[8] introduce two theorems that state that there exists a perfect discriminator if $\text{supp}(p_D)$ and $\text{supp}(p_G)$ are disjoint or if p_D and p_G are non-continuous (which they also prove).

- Given two distributions p_D and p_G with supports contained on disjoint manifolds $\mathcal{M}$ and $\mathcal{P}$ respectively, then there exists an optimal discriminator $D^* : X \rightarrow [0, 1]$ that is smooth in these supports, has accuracy 1, and $\nabla_x D^*(x) = 0 \forall x \in \mathcal{M} \cup \mathcal{P}$.
- Given two distributions p_D and p_G with supports contained on disjoint manifolds $\mathcal{M}$ and $\mathcal{P}$ that don't perfectly align and don't have full dimension, assuming the distributions are continuous in their respective manifolds, then again, there exists an optimal discriminator $D^* : X \rightarrow [0, 1]$ that has accuracy 1 in $\mathcal{M} \cup \mathcal{P}$, $\nabla_x D^*(x) = 0$, and $D*$ is smooth in these manifolds.

An accuracy of 1 means that the discriminator perfectly distinguishes between the samples from the dataset and the samples generated by the generator – i.e. it takes a value of 1 for samples in $\text{supp}(p_D)$ and a value of 0 for those in $\text{supp}(p_G)$. These theorems rely on the fact that p_G is non-continuous and that there isn't a perfect match between $\text{supp}(p_D)$ and $\text{supp}(p_G)$. This is shown with the following lemmas.

- If $G : Z \rightarrow X$ is a function that resembles a neural network (comprising of affine transformations and pointwise non-linearities) and the dimension of Z is lower than that of X then it is impossible for p_G to be continuous in the whole space (Lemma 1 in [8]).
- Given that p_D and p_G are continuous in the manifolds of their respective supports, it is increasingly unlikely that $\text{supp}(p_D)$ and $\text{supp}(p_G)$ perfectly align such that no arbitrarily small perturbation can dissatisfy this property (Lemmas 2 and 3 in [8]).

Having shown these theorems, [8] show that the gradient of the generator's objective in a standard GAN rapidly vanishes (Theorem 2.4). Further, they also show that when the generator objective is

modified to avoid this vanishing, the gradient of the log discriminator tends to have infinite variance (Theorem 2.6).

Both these results explain the degrading generator updates and instability in GAN training from a theoretical point of view. However, because the generator (policy) update procedure in adversarial IL is based on policy gradient methods, it is independent of the gradient of the discriminator (reward function in adversarial IL), meaning that the results on the generator's gradients are inapplicable to adversarial IL. In any case, the perfect discriminator theorems still apply to the discriminator in adversarial IL. This is because the generator is indeed a neural network mapping low-dimensional samples (features $s \in S$) to the discriminator's high dimensional input space (the space X comprising state transitions), albeit with an arbitrary function $W()$ acting on the generator's outputs. Further, p_G and p_D are potentially disjoint and it is at least unlikely that their supports perfectly align. Hence, even though the high-level mechanisms are different, adversarial IL is still inherently prone to optimisation-related training challenges.

A.3 Experiments on Optimisation Challenges in AIL

This section presents several experimental analyses that substantiate the theoretical reasoning in Section 2.0.1. Each set of experiments is elaborated in its own subsection.

A.3.1 Perfect Discriminator

To test the perfect discriminator hypothesis, we conduct the experiments from [8] on adversarial IL (Figure 8). We train adversarial motion priors [3] on a humanoid motion imitation task using the loss function from Equation (1) for the discriminator and Proximal Policy Optimisation (PPO) [11] to train the policy. Training is continued normally until some cut-off point. Then, with the policy updates paused, we retrain a discriminator to distinguish between samples in the expert dataset $\mathcal{M}$ and samples in p_G. The cut-off point is varied across runs to obtain varying levels of intersection between $\text{supp}(p_D)$ and $\text{supp}(p_G)$. Our experiments reproduce the results from [8] on adversarial IL. The discriminator loss from Equation (1) rapidly declines indicating a near-perfect discriminator prediction and highlighting the fact that even after sufficient training, p_G and p_D are non-continuous. The accuracy of the discriminator reaches a value of 1.0 in at most 75 iterations and $\nabla_x D(x)$ rapidly declines to be 0, further corroborating the theoretical results. Finally, we also find that the discriminator's predictions on the motions generated by the policy rapidly drop down to zero, meaning that the policy receives unhelpful updates.

A.3.2 High Discriminator Variance

To verify the high variance hypothesis, we use the same experimental setup as before but now train AMP for longer to obtain even more intersection between $\text{supp}(p_D)$ and $\text{supp}(p_G)$. The discriminator is slightly modified by removing the sigmoid activation at the output layer and instead computing the loss on $\texttt{sigmoid}(D())$ [5]. Here, instead of retraining the discriminator after the cut-off point, we continue its training to maintain the learnt decision boundary and visualise the variance in the trained discriminator's predictions on the motions generated by an unchanging policy. We hypothesise that as training continues and the supports of the two distributions get closer, the discriminator is less likely to see samples from a region outside $\text{supp}(p_D) \cup \text{supp}(p_G)$, meaning that its variance reduces as training progresses. We observe that the discriminator's predictions indeed have quite a high variance and the range of the predictions varies vastly across training levels (Figure 9). Further, the variance indeed reduces over the training level, indicating a gradually increasing intersection between $\text{supp}(p_D)$ and $\text{supp}(p_G)$. The adversarial optimisation is likely to get stabilised as the policy gets closer to optimality, however, training for the most part is still rather unstable because of the high variance in the discriminator's predictions.

A.3.3 Discriminator Non-Smoothness

Finally, we carry out experiments to understand the smoothness of the learnt reward function as well as its changes over training iterations (Figure 10). To do this, we again train AMP on a humanoid

[5] This is done to allow the network to predict any arbitrary value and to allow more flexibility in the reward function transformation. The same is done in the original AMP procedure [3] and we make no additional modifications to their code.

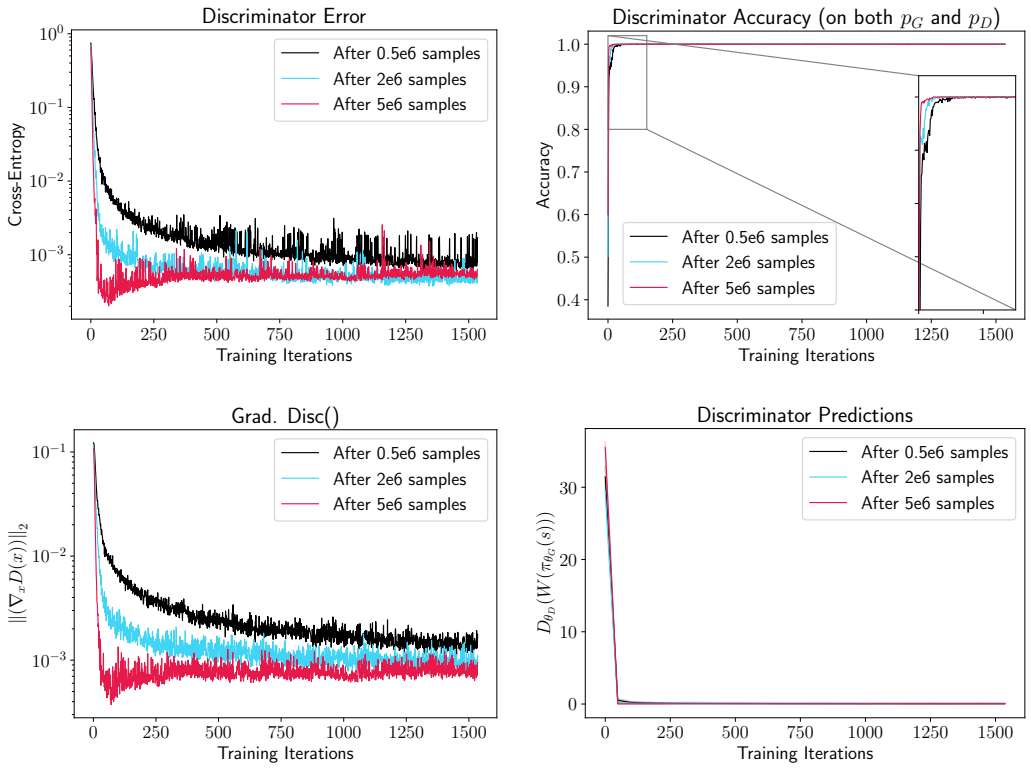

Figure 8: Perfect discriminator experiments. The policy was first trained for $0.5e6$, $2e6$, and $5e6$ data samples. Then, with the policy updates paused, the discriminator was retrained. We find that the discriminator very quickly learns to perfectly distinguish between the motions in $\mathcal{M}$ and the motions produced by the policy (notice the logarithmic scale).

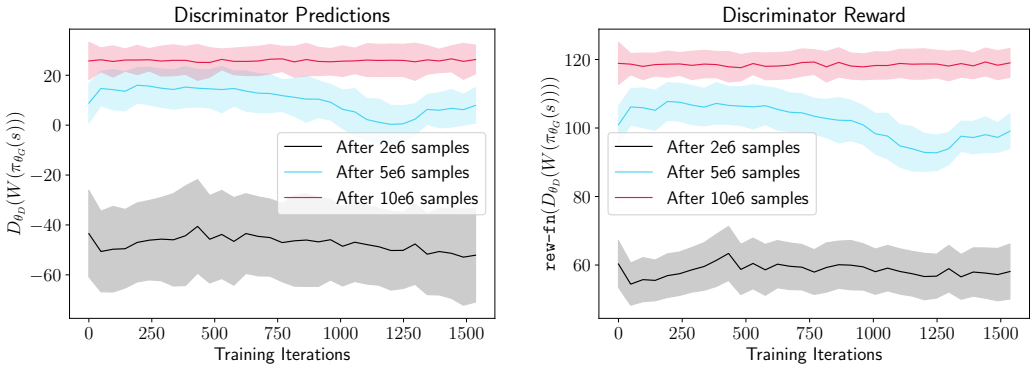

Figure 9: Discriminator variance experiments. The policy was first trained for $2e6$, $5e6$, and $10e6$ data samples. Then, with the policy updates paused, discriminator training was continued. We observe high variance in the rewards received. Given that the policy is unchanging, a high variance in the reward indicates poor reinforcement learning.

walking task. This time we do not make any modifications to the algorithm and simply evaluate the discriminator at gradually increasing distances from the true data manifold at various points in training. We find that the discriminator's predictions on average, decline as we move farther away from the true data manifold. However, again, the predictions are quite noisy and have a fairly large standard deviation.

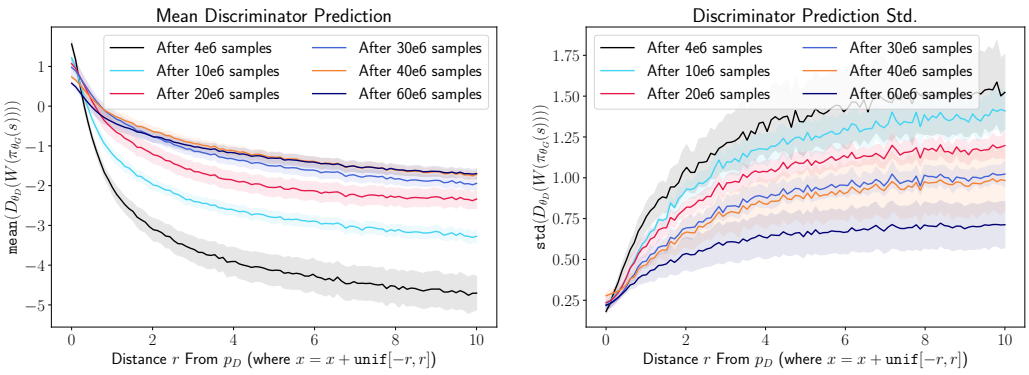

Figure 10: Discriminator non-smoothness experiments. Plots show the mean and std. discriminator prediction over a large batch of perturbed expert data samples. Notice the high value of the std. compared to the mean at any given distance from p_D.

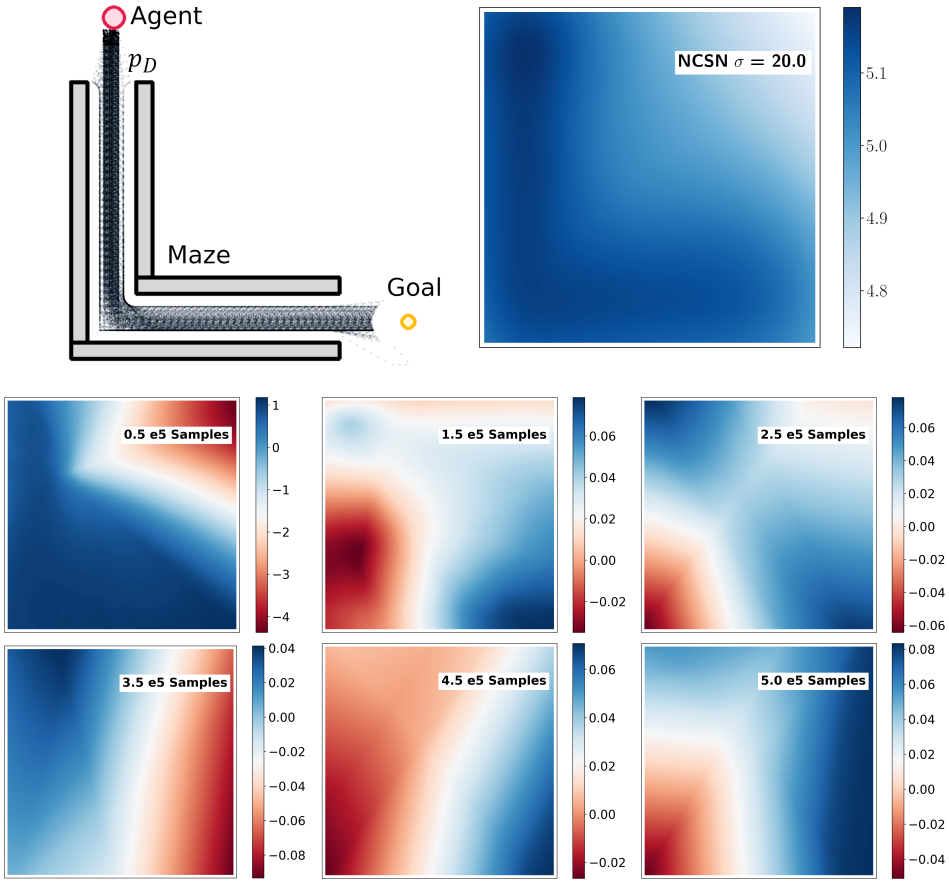

Figure 11: Top Left: Structure of the 2D environment. Top Right: Rewards learnt using NEAR. Bottom: Various stages of the AMP discriminator during learning.

To get a better visual understanding of the discriminator's behaviour, we also conduct the same experiment in a 2D imitation problem (where the agent's state is its 2D position) shown in Figure 11. Here, the agent is a spherical object whose goal is to reach the target at the bottom-right corner. Expert demonstrations were collected such that the expert's trajectory does not reach the target directly but instead first passes through an L-shaped maze. It is expected that the agent learns to imitate this by

also passing through the maze. We visualise the discriminator's predictions on the 4D state transition features by plotting the mean prediction in the agent's reachable set at every point in a 2D discredited grid. AMP is trained on this simple task for $0.5e6$ training samples and the discriminator's predictions are plotted at several points in training. The experiment visually demonstrates the non-smooth and non-stationary nature of the discriminator. In contrast, we also train our proposed method on this task and see that the learnt reward function is smooth and very closely resembles the expert's data distribution.

B Tasks

The task reward and goal features for each imitation task are described below.

Target Reaching

In this task, the agent's objective is to navigate towards a randomly placed target. The agent's state is augmented to include a goal $g_t = x_t^*$ where t is the current timestep, and x_t^* is the target's position in the agent's local coordinate frame. During training, the target is randomly initialised in a 240° arc around the agent within a radius in the range [2.0, 6.0] meters. The agent is rewarded for minimizing its positional error norm and heading error to the target. Here, x_t is the agent's position, v_t is the agent's velocity vector, $d^*{}_t$ is a unit vector pointing from the agent's root to the target, and v^* is a scalar desired velocity set to 2.0 for walking and 4.0 for runninng.

$$r^{task} = 0.6(\exp\left(-0.5\left\|x_t^* - x_t\right\|^2\right)) + 0.3(1 - \frac{2}{1+\exp\left(5 * \frac{(v_t * d^*{}_t)}{\|v_t\|}\right)}) + 0.1(1 - (\|v_t\| - v^*)^2) \quad (5)$$

Target Reaching & Punching

In this task, the agent's objective is to both reach a target and then strike it with a left-handed punch. Here, the goal is a vector of the target's position in the agent's local frame and a boolean variable indicating whether the target has been punched, $g_t =< x_t^*, \texttt{punch state} >$. We use the same target initialisation strategy as before with an arc of 45° and an arc radius in [1.0, 5.0] meters. The agent is rewarded using the target location reward when it is farther than a threshold distance from the target and with a target striking reward when it is within this threshold. The striking reward aims to minimise the pose error and heading error between the agent's end effector and the target while simultaneously aiming to achieve a certain end effector velocity and height. The complete reward function is shown below where x_t^{eff} is the end effector position, v_t^{eff} is the end effector velocity vector, h_t^{eff} is the end effector height, and $v^{*eff} = 4.0$ and $h^{*eff} = 1.4$ are scalar desired punch speed and height.

$$r^{task} = \begin{cases} 1.0 & \text{target has been hit} \\ r^{near} & \left\|x_t - x_t^*\right\| < 1.2 \\ r^{far} & \text{otherwise} \end{cases} \quad (6)$$

$$\begin{aligned} r^{far} &= \text{Equation (5)} \\ r^{near} &= 0.3 + 0.3(0.1(\exp\left(-2.0\left\|x_t^* - x_t^{eff}\right\|^2\right)) + 0.4(1 - \frac{2}{1+\exp\left(5 * \frac{(v_t^{eff} * d^*{}_t)}{\left\|v_t^{eff}\right\|}\right)}) \\ &\quad + 0.3(1 - (\left\|v_t^{eff}\right\| - v^{*eff})^2) + 0.2(1 - (h_t^{eff} - h^{*eff})^2)) \end{aligned} \quad (7)$$

Unconditioned Rewards

These reward functions were used in our ablation experiments. In this case, we do not use any additional goal-conditioning.

Walking & Running: The agent is rewarded positively for every time step in the episode, encouraging longer episode lengths. Further, the agent is also rewarded for relative positive dis-

placement, with the clipping at 0.5 meters. The final task reward is a weighted combination of both. $r_t^{task} = 0.5 \cdot 1 + 0.5 \cdot \frac{\texttt{clip}(x_t - x_{t-1}, 0.0, 0.5)}{0.5}$.

Crane Pose & Punching: The agent is rewarded positively for every time step in the episode, encouraging longer episode lengths. $r_t^{task} = 1$.

C Training & Evaluation Details

C.1 Architectures

For both NEAR and AMP, the policy is a simple feed-forward neural network that maps the agent's state s to a Gaussian distribution over actions, $\pi_{\theta_G} = \mathcal{N}(\mu(s), \Sigma)$ with the mean $\mu(s)$ being returned by the neural network and a fixed diagonal covariance matrix Σ. In our experiments, the neural network is a fully-connected network with (1024, 512) neurons and ReLU activations. Σ is set to have values of $e^{-2.9}$ and stays fixed throughout training. The critic (value function) is also modelled by a similar network. The value function is updated with TD(λ) [26] and advantages are computed using generalised advantage estimation [27]. When using the environment-supplied task reward, we set $w^{task} = w^{energy} = 0.5$.

The NCSN neural network is a fully-connected network with an auto-encoder style architecture. Here, the encoder has (512, 1024) neurons and maps the input to a 2048-dimensional latent space. The decoder has (1024, 512, 128) neurons with the output being the unconditional energy of a sample. We use ELU activations between all layers of the auto-encoder and use Xavier uniform weight initialisation [28] to improve consistency across different independent training runs. Further, we standardise samples before passing them to the network. The NCSN noise scale was defined as a geometric sequence with $\sigma_1 = 20$, $\sigma_L = 0.01$, and $L = 50$ following the advice from [16]. Following [16] also track the exponentially moving average (EMA) of the weights of the energy network during training and use the EMA weight during inference, as it has been shown to further reduce the instability in the sample quality. All models in this paper were trained on the Nvidia-A100 GPU [29; 30].

C.2 Reinforcement Learning

We borrow the experimental setup from [3] where the agent's state is a 105-dimensional vector consisting of the relative position of each link with respect to the root body and the rotation of each link (represented as a 6-dimensional normal-tangent vector of the link's linear and angular velocities). All features are in the agent's local coordinate system. Similarly to [3], we do not add additional features to encode information like the feature's phase in the motion, or the target pose. Further, the character is not trained to replicate the phase-wise features of the motion and the learnt rewards are generally only a representation of the closeness of the agent's motion to the expert's data distribution. The agent's actions specify a positional target that is then tracked via PD controllers at each joint. The expert's motions are obtained from the CMU and SFU motion capture datasets and contain trajectories of several motions. For each motion, a dataset of state transitions $\mathcal{M} \equiv \{(s, s')\}$ is created to learn an imitation policy.

We use asynchronous parallel training in the IsaacGym simulator [31] for all experiments and analyses in this paper and spawn 4096 independent, parallel environments during both training and evaluation. Given an initial policy, training is carried out by first rolling out the policy in all environments for a rollout horizon (16 in our experiments). During rollouts, an environment is reset if it happens to reach the `done` state. A replay buffer is then populated with all agents' transitions obtained from the rollouts. Then, with rollouts paused, the policy and value function are updated with the data from the replay buffer. After the update, the rollouts are restarted with the updated policy. We use multiple mini epochs to update the policy and value function at every update step. In each mini epoch, several mini-batches of data samples are drawn from the replay buffer. The number of mini-batches depends on the relative size of each mini-batch and the replay buffer [6] In the experiments on the AMP discriminator, we used $N_{envs} = 1024$, meaning that 500k training steps correspond to about 1464 iterations. In this setup, 500k samples was also approximately the training needed to see

[6] Given a horizon h, $N_{envs} = 4096$, $N_{minibatches} = 8$, and $N_{miniepochs} = 6$, the number of training iterations for N training steps is $\frac{N}{h \cdot N_{envs}} \cdot N_{minibatches} \cdot N_{miniepochs}$.

any noticeable improvement in the learnt policy and was hence also chosen as a training bound to determine the variance in the discriminator's predictions.

During policy evaluation, the same rollout procedure is used, however this time, the rollout horizon is set to 300 and the networks are of course not updated. Performance metrics are recorded every main training epoch as a mean across the $k = 20$ most rewarding environments. In the experiments on the AMP discriminator, the standard deviation across these environments was used to plot the error regions. Finally, we use reference state initialisation to initialise all environments at a random state in the expert's motion dataset use early termination to reset when the agent falls over. For certain tasks like spin-kick, we find that it is especially challenging to learn a policy starting from certain initial states (such as the jumping-off point). Additional exploration is required to learn the optimal actions starting from these states. In these cases, reference states are drawn from a beta distribution instead of a uniform distribution (with $\beta = 3.0$ and $\alpha = 1.0$). For temporally composed tasks like target reaching and punching, we use both reference motions during initialisation. In this case, both reference motions are used with a probability of 0.5 and the target is initialised in the range [1.0, 1.2] when the agent is initialised with the punching reference.

C.3 Evaluation Metrics

Average Dynamic Time Warping Pose Error: This is the mean dynamic time warping (DTW) error [22] between trajectories of the agent's and the expert's poses averaged across all expert motions in the dataset. Given a set of j expert motion trajectories $\hat{\tau}_j$ of arbitrary length L_j where each trajectory is a series containing the Cartesian positions of the reference character's joints $\hat{x}_i$, $D = \{\hat{\tau}_j\}_{j=1}^{N_{traj}}$ where $\hat{\tau}_j = \{\hat{x}_i\}_{i=1}^{L_j}$, first, we roll out the trained policy deterministically [7] across several thousand random starting-pose initialisations [8]. Then, the $k = 20$ most rewarding trajectories are selected to form a set of policy trajectories $D_\pi = \{\tau_m\}_{m=1}^{k}$ where $\tau_m = \{x_i\}_{i=1}^{L_m}$ and each trajectory has an arbitrary length L_m. The average dynamic time warping pose error is then computed as the average DTW score of all τ_m across all expert trajectories $\hat{\tau}_j$ with $\|\hat{x}_i - x_i\|_2$ as the cost function. To ensure that the pose error is only in terms of the character's local pose and not its global position in the world, we transform each Cartesian position to be relative to the character's root body position at that timestep ($\hat{x}_i \leftarrow \hat{x}_i - \hat{x}_i^{root}$ and $x_i \leftarrow x_i - x_i^{root}$).

Spectral Arc Length: Spectral Arc Length (SAL) [23; 24; 25] is a measure of the smoothness of a motion. The smoothness of the character's trajectory is an interesting metric to determine the policy's ability to perform periodic motions in a controlled manner. The underlying idea behind SAL is that smoother motions typically change slowly over time and are comprised of fewer and low-valued frequency domain components. In contrast, jerkier motions have a more complex frequency domain signature that consists of a lot of high-frequency components. The length of the frequency domain signature of a motion is hence an appropriate indication of a motion's smoothness (with low values indicating smoother motions). SAL is computed by adding up the lengths of discrete segments (arcs) of the normalised frequency-domain map of a motion. In our experiments, we use SPARC [23], a more robust version of the spectral arc length that is invariant to the temporal scaling of the motion. We track the average SAL of the $k = 20$ most rewarding trajectories generated by the policy at different training intervals and use the root body Cartesian coordinates to compute the SAL. Note that in this case, we do not transform the positions to the agent's local coordinate system.

C.4 Repeatability & Determinism

Each algorithm was trained 5 times independently on every task with separate random number generator seeds for each run. However, using a fixed seed value will only potentially allow for deterministic behaviour in the IsaacGym simulator. Due to GPU work scheduling, it is possible that runtime changes to simulation parameters can alter the order in which operations take place, as environment updates can happen while the GPU is doing other work. Because of the nature of floating point numeric storage, any alteration of execution ordering can cause small changes in the least significant bits of output data, leading to divergent execution over the simulation of thousands of environments and simulation frames. This means that experiments from the IsaacGym simulator (including the original work on AMP) are not perfectly reproducible on a different system. However,

[7] Deterministic here means that we use the predicted mean as the action instead of sampling from $\mathcal{N}(\mu(x), \Sigma)$.

[8] This is done to ensure that the computed performance is not biased to any single initial state.

parallel simulation is a major factor in achieving the results in this paper and minor non-determinism between independent runs is hence just an unfortunate limitation. More information on this can be found in the IsaacGymEnvs benchmarks package. Note that this is only a characteristic of the reinforcement learning side of our algorithm. The pretrained energy functions are also seeded and these training runs are perfectly reproducible.

D Extended Experiment Results

Table 4: A comparison of the mean performance of NEAR and AMP at the end of training (*Avg. pose error:* lower is better. *Others:* closeness to expert is better).

Task	Num. Motion Clips	Algorithm	Avg. Pose Error (m)	Spectral Arc Length	Root Body Velocity ($\frac{m}{s}$)	Root Body Jerk ($\frac{m}{s^3}$)
Walking	74 (249.38 sec.)	NEAR	**0.51**	**-7.52**	**1.25**	**360.89**
		AMP	**0.51**	-8.78	1.87	736.32
		Expert's Value		-5.4	1.31	130.11
Running	26 (32.55 sec.)	NEAR	**0.62**	**-7.24**	**3.52**	**1298.42**
		AMP	0.65	-9.71	3.79	1560.14
		Expert's Value		-3.79	3.55	513.68
Crane Pose	3 (24.62 sec.)	NEAR	0.94	-6.6	0.12	**46.77**
		AMP	**0.82**	**-8.1**	**0.03**	19.29
		Expert's Value		-12.28	0.03	49.05
Left Punch	19 (4.91 sec.)	NEAR	0.37	**-6.87**	0.01	11.34
		AMP	**0.32**	-9.93	**0.06**	**29.6**
		Expert's Value		-1.73	0.16	72.49
Mummy Walk	1 (5.40 sec.)	NEAR	0.66	**-4.72**	0.33	**189.73**
		AMP	**0.41**	-13.84	**0.98**	354.49
		Expert's Value		-4.71	0.73	79.63
Spin Kick	1 (1.15 sec.)	NEAR	0.78	-5.59	**0.53**	286.63
		AMP	**0.58**	**-3.16**	0.5	**278.25**
		Expert's Value		-3.39	1.05	273.61

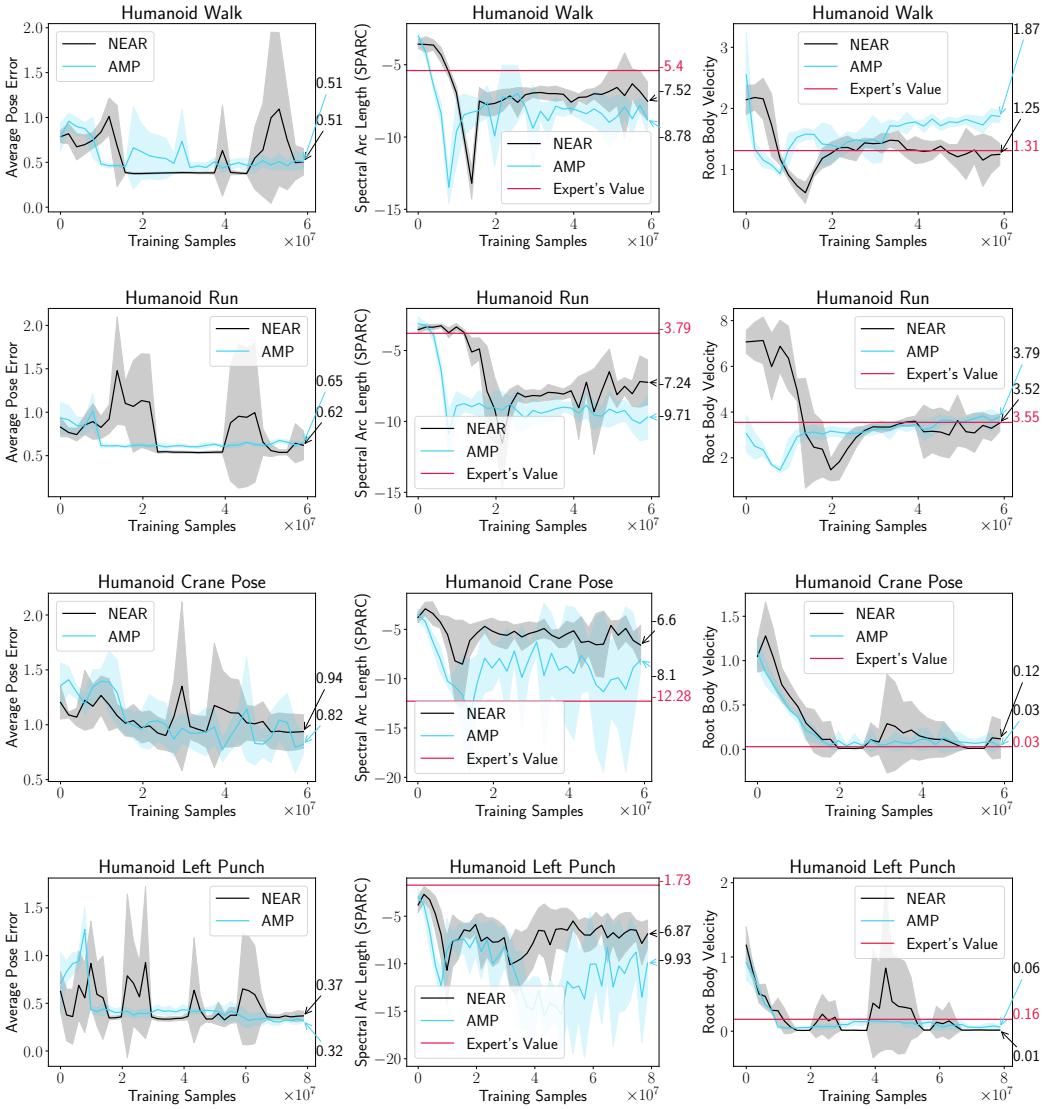

Figure 12: An extended set of performance metrics recorded in our experiments.

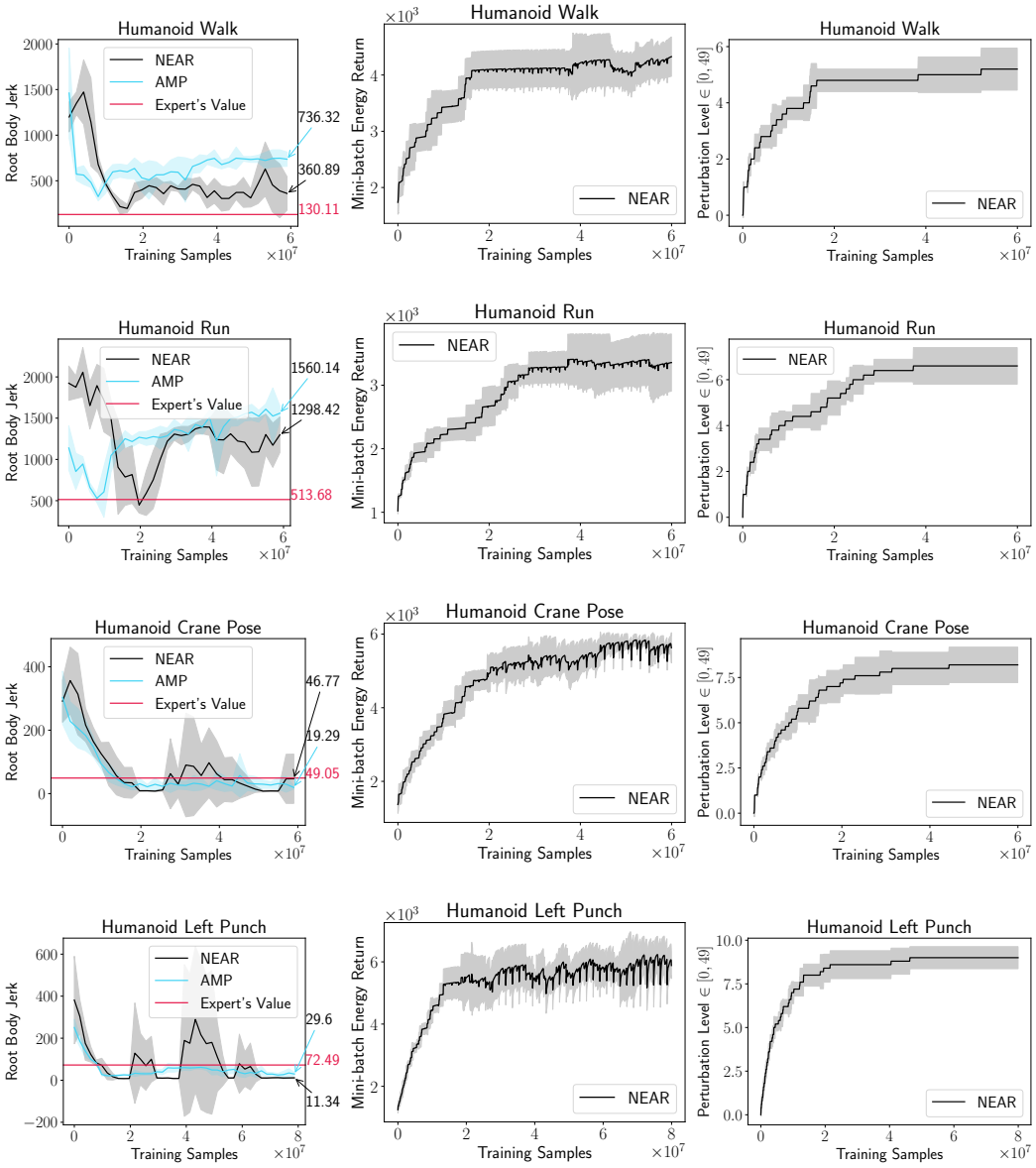

Figure 13: An extended set of performance metrics recorded in our experiments.

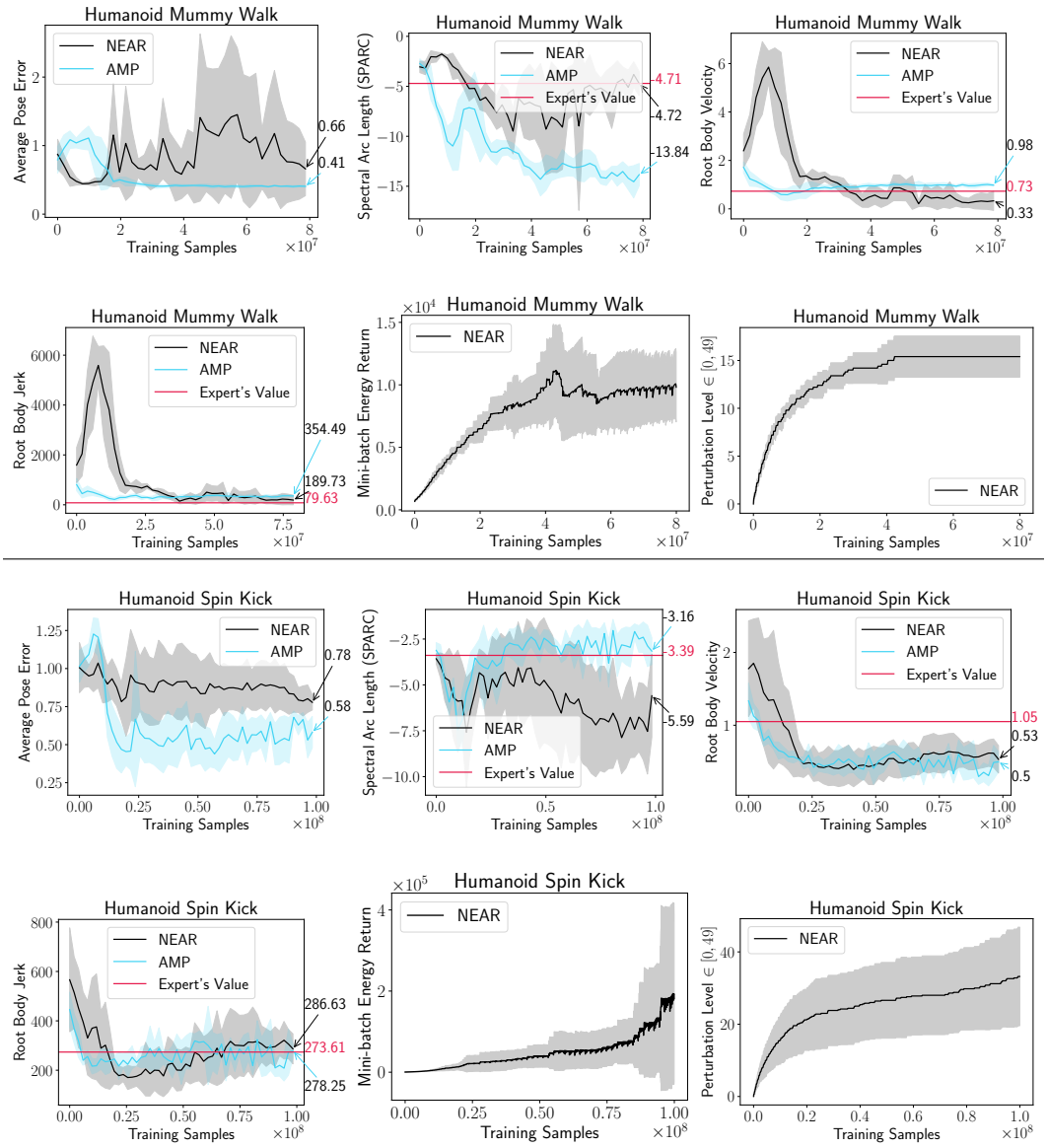

Figure 14: An extended set of performance metrics recorded in single-clip imitation tasks.

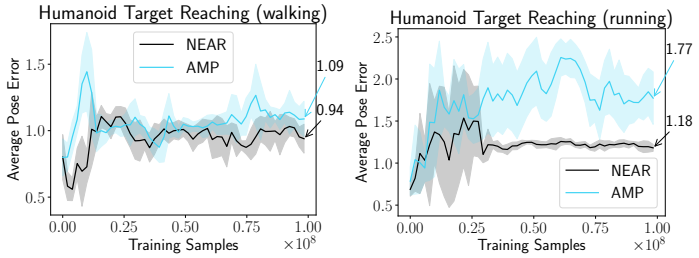

Figure 15: An extended set of performance metrics recorded in the experiments on goal-conditioned learning with composed reward functions.

E Extended Ablations

Table 5: Task reward composition ablation with annealing.

Task	Ablated Configuration	Avg. Pose Error (m)	Spectral Arc Length	Root Body Velocity ($\frac{m}{s}$)	Root Body Jerk ($\frac{m}{s^3}$)
Walking	anneal & e_θ	0.51	-7.52	**1.25**	360.89
	anneal & $\tilde{r}$	**0.42**	**-6.11**	0.88	**249.97**
	Expert's Value		-5.4	1.31	130.11
Running	anneal & e_θ	0.62	**-7.24**	**3.52**	**1298.42**
	anneal & $\tilde{r}$	**0.59**	-8.02	4.86	1875.44
	Expert's Value		-3.79	3.55	513.68
Crane Pose	anneal & e_θ	**0.94**	-6.6	**0.12**	**46.77**
	anneal & $\tilde{r}$	1.33	**-7.24**	0.14	72.24
	Expert's Value		-12.28	0.03	49.05

Table 6: Task reward composition ablation without annealing.

Task	Ablated Configuration	Avg. Pose Error (m)	Spectral Arc Length	Root Body Velocity ($\frac{m}{s}$)	Root Body Jerk ($\frac{m}{s^3}$)
Walking	σ_5 & e_θ	0.49	**-7.1**	1.58	653.17
	σ_5 & $\tilde{r}$	**0.42**	-8.7	**1.25**	**360.86**
	Expert's Value		-5.4	1.31	130.11
Running	σ_5 & e_θ	0.62	**-6.69**	**3.48**	**1334.1**
	σ_5 & $\tilde{r}$	**0.57**	-8.61	4.76	1826.98
	Expert's Value		-3.79	3.55	513.68
Crane Pose	σ_5 & e_θ	1.38	**-6.03**	0.07	**29.5**
	σ_5 & $\tilde{r}$	**1.23**	-4.34	**0.02**	16.3
	Expert's Value		-12.28	0.03	49.05

Table 7: Annealing ablation with learnt energy reward.

Task	Ablated Configuration	Avg. Pose Error (m)	Spectral Arc Length	Root Body Velocity ($\frac{m}{s}$)	Root Body Jerk ($\frac{m}{s^3}$)
Walking	anneal & e_θ	0.51	-7.52	**1.25**	**360.89**
	σ_5 & e_θ	**0.49**	**-7.1**	1.58	653.17
	Expert's Value		-5.4	1.31	130.11
Running	anneal & e_θ	**0.62**	-7.24	**3.52**	**1298.42**
	σ_5 & e_θ	**0.62**	**-6.69**	3.48	1334.1
	Expert's Value		-3.79	3.55	513.68
Crane Pose	anneal & e_θ	**0.94**	**-6.6**	0.12	**46.77**
	σ_5 & e_θ	1.38	-6.03	**0.07**	29.5
	Expert's Value		-12.28	0.03	49.05

Table 8: Annealing ablation with composed task reward.

Task	Ablated Configuration	Avg. Pose Error (m)	Spectral Arc Length	Root Body Velocity ($\frac{m}{s}$)	Root Body Jerk ($\frac{m}{s^3}$)
Walking	anneal & $\tilde{r}$	**0.42**	**-6.11**	0.88	**249.97**
	σ_5 & $\tilde{r}$	**0.42**	-8.7	**1.25**	360.86
	Expert's Value		-5.4	1.31	130.11
Running	anneal & $\tilde{r}$	0.59	**-8.02**	4.86	1875.44
	σ_5 & $\tilde{r}$	**0.57**	-8.61	**4.76**	**1826.98**
	Expert's Value		-3.79	3.55	513.68
Crane Pose	anneal & $\tilde{r}$	1.33	**-7.24**	0.14	**72.24**
	σ_5 & $\tilde{r}$	**1.23**	-4.34	**0.02**	16.3
	Expert's Value		-12.28	0.03	49.05

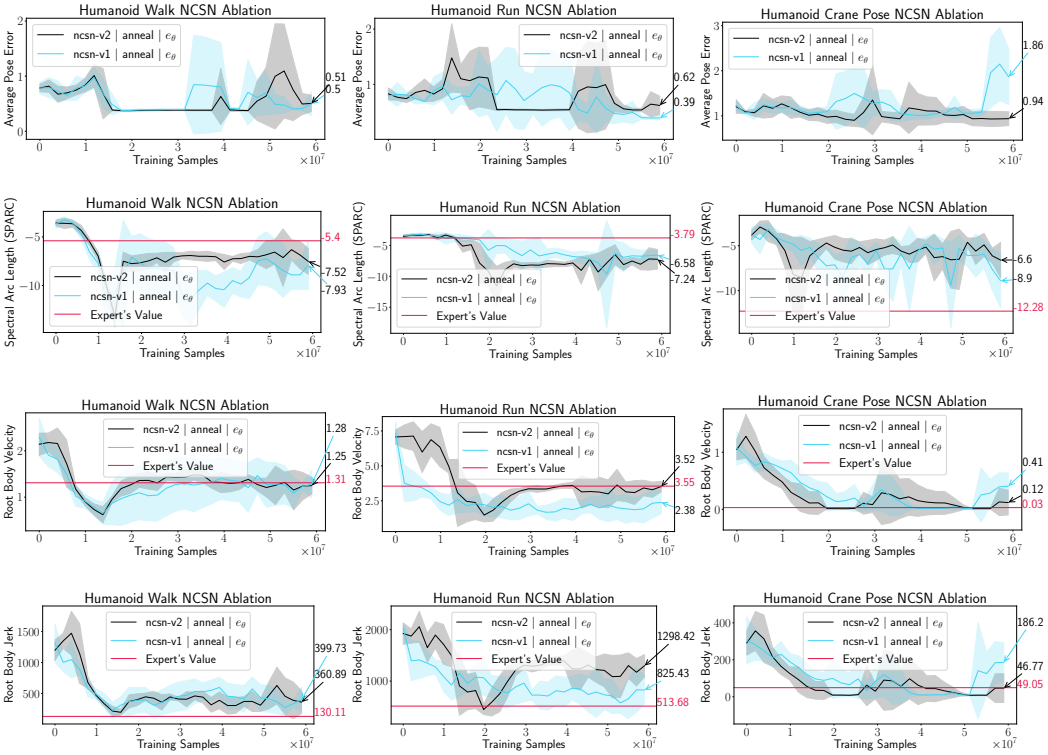

Figure 16: An extended set of NCSN ablations.