NEURAL PROBABILISTIC MOTOR PRIMITIVES FOR HUMANOID CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

Transferring functional properties from one or multiple expert policies to a student policy is an important challenge in control. Expert robustness is of particular interest; we would like to not only transfer the expert behavior but also its ability to recover from perturbations. With this in mind, we explore approaches for *policy cloning* and propose *linear feedback policy cloning* as a simple option for certain settings. We show that it can be surprisingly straightforward to clone expert policies for seemingly complex behaviors without the student requiring any environment interactions. We then propose a latent-variable architecture that bottlenecks a sensory-motor primitive space, which, again, can be trained entirely offline to compress thousands of expert policies. We show this resulting *neural probabilistic motor primitive* system produces robust one-shot imitation of whole-body humanoid behaviors. In addition, we analyze the resulting latent space and demonstrate the ability to reuse this system. We encourage readers to view a supplementary video summarizing our results.

1 INTRODUCTION

A broad challenge in machine learning for control and robotics is to produce policies capable of general, flexible, and adaptive behavior of complex, physical bodies. To build policies that can effectively control simulated humanoid bodies, researchers must simultaneously overcome foundational challenges related to high-dimensional control, body balance, and locomotion. Recent progress in deep reinforcement learning has raised hopes that such behaviors can be learned end-to-end with minimal manual intervention. Yet, even though significant progress has been made thanks to better algorithms, training regimes, and computational infrastructure, the resulting behaviors still tend to exhibit significant idiosyncrasies (e.g. Heess et al., 2017; Bansal et al., 2018).

One advantage of working with humanoids in this context is that motion capture data is widely available and can serve to help design controllers that produce apparently humanlike movement. Indeed, recent developments are now allowing for the production of highly specialized expert policies which robustly, albeit narrowly, reproduce single motion capture clips.

A remaining challenge on the way to truly flexible and general purpose control is to be able to sequence and generalize individual skills in a task-directed manner. Achieving this goal requires not just the ability to acquire individual skills in the first place, but also an architecture and associated training procedure that then allows to represent, recruit, and compose such a large number of skills in a robust manner with as little additional tuning as possible.

This paper presents a step in this direction. Specifically, the setting we focus on will be one in which we have a large number of robust experts that perform single skills well and we wish to transfer these skills into a shared policy that can do what each expert does as well as the expert, while also generalizing to unseen behaviors within the distribution of skills. The desiderata in designing our approach were to enable robust control for diverse skills in a form that allowed for one-shot imitation as well as permitting straightforward reuse of skills in a task-directed manner. Furthermore, we would like our approach to scale to a very large number of individual skills while also minimizing the level of manual intervention that is required to enable their composition and sequencing.

To this end, we present two contributions. Our first contribution is a scalable proposal for offline transfer and compression of expert skills. The core intuition underlying this approach is that we

often have a sense of the relevant set of states at which we wish to copy expert behavior, and in such settings we should be able to comprehensively transfer the functional properties of an expert to a student policy without having to engage in RL-style closed-loop training. We call the functional transfer of policy content *policy transfer* or *policy cloning*, and term our specific proposal *linear feedback policy cloning (LFPC)*. Avoiding closed-loop RL training allows our approach to work with thousands of skills at the same time. Our second contribution is the development of a neural network architecture for representing a space of probabilistic motor primitives, which we refer to as *neural probabilistic motor primitives*. This architecture is designed to perform one-shot imitation, while learning a dense embedding space of a large number of individual motor skills. Once trained, this module does not just reproduce individual behaviors in the training data, but allows to sequence and compose these behaviors in a controlled fashion, and synthesis of novel movements consistent with the training data distribution.

1.1 Background & Related Work

Recent efforts in RL for *humanoid control* build on a large body of research in robotics and animation. While contemporary results for learning from scratch (Schulman et al., 2015; Heess et al., 2017) can be impressive the behaviors are not consistently human-like. Learning from motion capture (mocap) can provide strong constraints and several recent approaches have demonstrated that it is possible to acquire specific behavioral skills, possibly jointly with external RL objectives (Merel et al., 2017; Peng et al., 2018; Liu & Hodgins, 2018). At present, the policies produced tend to be restricted to single skills/behaviors and can require very large quantities of environment interactions, motivating us to seek methods which reuse existing single-skill expert policies.

Knowledge transfer refers to the broad class of approaches which transfer the input-output functional mapping, to some extent or another, from a *teacher* (or *expert*) to a *student* (Hinton et al., 2015; Srinivas & Fleuret, 2018; Furlanello et al., 2018). *Distillation* connotes the transfer of function from one or more expert systems into a single student system often with the goal of compression or of combining multiple experts qualities (Hinton et al., 2015; Parisotto et al., 2015; Rusu et al., 2015; Teh et al., 2017). *Imitation learning* is the control-specific term for the production of a student policy from either an expert policy or the behavioral demonstrations of an expert. One basic algorithm is *behavioral cloning*, which refers to supervised training of the policy from state-action pairs. In the most simple case it only requires examples from the expert. A broader setting is that in which more liberal queries to the expert are permitted; e.g. for the online-imitation setting as in DAGGER (Ross et al., 2011). This setting is often satisfied e.g. if we wish to combine behavior from multiple experts.

One-shot imitation is a concept which means that a trained system, at test time, can watch an example behavior and imitate it, as, for instance, in Duan et al. (2017). More similar to our work is the setting examined by Wang et al. (2017), in which full-body humanoid movements were studied. Compared with this latter work, we will employ an architecture here that encourages imitation of motor details, rather than overall movement type, and we scale our approach to more expert demonstrations.

The notion of *motor primitives* is widespread in neuroscience, where there is evidence that lower dimensional control signals can selectively coordinate and blend behaviors which in some cases are built into the spinal system (Bizzi et al., 2008), or that the cortex organizes the behavioral space of primitive motor behaviors (Graziano, 2006). In this work a motor primitive reflects a context triggered execution of a self-stabilized motor pattern (i.e. a robust policy execution in an environment), and the particular architecture we are considering is inspired by the formalization presented in Todorov & Ghahramani (2003), which places a probabilistic latent bottleneck on the sensory-motor mapping. Note that this notion is related but distinct to the line of research into "movement primitives" (Schaal et al., 2003; Neumann et al., 2014) in the robotics literature. These "movement primitives" can be seen as a particular implementation choice for a motor primitive, which emphasize parametrization and learning of target trajectories rather than learning the control-stabilization element, which is often handled by a pre-specified PID controller.

It has previously been recognized that linear-feedback policies can work well around optimal trajectories or limit cycles even for high DoF bodies. These can be obtained by sample-based optimization (e.g. Ding et al. (2015)) or by differential dynamic programming (Morimoto & Atkeson, 2003; Tassa et al., 2012; 2014). For linear-quadratic-Gaussian control (Athans, 1971) or differential dynamic programming (Mayne, 1966; Jacobson & Mayne, 1970), we obtain feedback policies

where the feedback terms are computed from the value function, amounting effectively to feedback-stabilized plans. Work by Mordatch et al. (2015) has shown that linear-feedback policies resulting from trajectory optimization can be used to train neural networks. We employ a similar idea to transfer optimal behavior from an existing policy, observing that an optimal policy implicitly reflects the structure of the (local) value landscape and appropriately functions as a feedback controller.

2 Transfer and compression of expert behaviors

2.1 Obtaining experts from motion capture data

In order to study how to transfer and consolidate experts, we must be able to generate adequate quantities of expert data. For this work, we use expert policies trained to reproduce motion capture clips. The approach we use for producing experts is detailed more fully in Anonymous (2019) and largely follows Peng et al. (2018). It yields time-indexed neural network policies that are robust to moderate amounts of action noise (see appendix A for addional details on the training procedure). Some examples of the resulting single-skill time-indexed policies that are obtained from this procedure are depicted in Fig. 1. All our experts were trained in Mujoco environments.

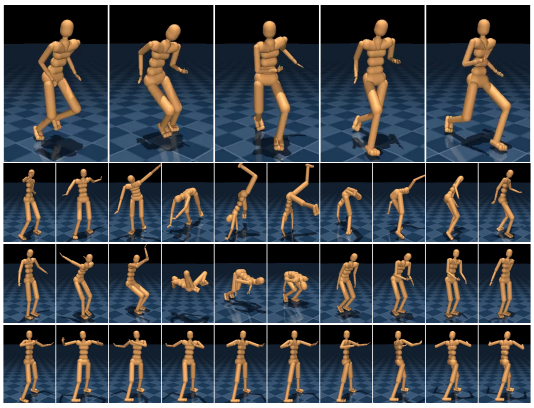


Figure 1: Examples of representative experts learned from motion capture. From top to bottom, these are "run and dodge", "cartwheel", "backflip", and "twist". See accompanying video. Note that these four behaviors will be used as representative examples for validation in single-skill transfer experiments.

Data We use the CMU Mocap database[1], which contains more than 2000 clips of varying lengths from more than 100 subjects. The motions in this dataset are quite varied, including many clips of walking, turning, running, jumping, dancing, various hand movements, and many more idiosyncratic behaviors. From this, we selected various clips of generic whole-body movements – any clips longer than 6 seconds were cut into smaller pieces yielding approximately 3000, roughly 2-6 second snippets. Just over half of these are generic locomotion such as walking, running, jumping and turning. The rest of the clips mostly contained diverse hand movements while standing. We trained one expert policy per selected snippet, yielding 2707 expert policies in our training set.

2.2 Training a student policy from a set of examples

When transferring knowledge from an expert policy to a student we would like the student to replicate the expert's behavior in the full set of states plausibly visited by the expert. In our case experts trained to reproduce single clips can be conceptualized as nonlinear feedback controllers around a nominal trajectory, and the manifold of states visited by experts can be thought of as a tube around that reference. We require the student to be able to operate successfully in and remain close to this tube even in the face of small perturbations.

Since we are aiming to compress the behavior of thousands of experts we desire a computationally efficient method. We investigate two schemes that allow us to record the experts' state-action mappings on a small-sample estimate of the experts' state distributions and to then train the student via supervised learning. Both schemes are convenient to implement in a regular supervised learning pipeline and require neither querying many experts simultaneously (which limits scalability when dealing with thousands of experts) nor execution of the student at training time.

[1]The CMU motion capture database is available at mocap.cs.cmu.edu.

Behavioral cloning from noisy rollouts The first approach amounts to simply gathering a number of noisy trajectories from the expert (either under a stochastic policy or with noise injection) while logging the optimal/mean action of the expert instead of the noisy action actually executed. A version of this is equivalent to the DART algorithm of Laskey et al. (2017). We then perform behavioral cloning from that data.

Linear-feedback policy cloning (LFPC) The second approach, which we refer to as linear-feedback policy cloning (LFPC), logs the action-state Jacobian as well as the expert action along a single nominal trajectory. The Jacobian can be used to construct a linear feedback controller which gives target actions in nearby perturbed states during training (described below).

Given an expert policy π_E, let $\mu_E(s)$ be the mean action of the expert in state s. The nominal trajectory refers to the sequence of nominal state-action pairs $\{s_t^\star, a_t^\star\}_{1\ldots T}$ obtained by executing $\mu_E(s)$ recursively from an initial point $s_0^\star$. As pointed out above, experts trained to reproduce single clips robustly can be thought of as nonlinear feedback controllers around this nominal trajectory. We observe that we can linearize the feedback policies of experts available as neural networks. Let δs be a small perturbation of the state and let $\boldsymbol{J} = \frac{d\mu_E(s)}{ds}|_{s=s}$ be the Jacobian. Then

$$\mu_E(s + \delta s) = \mu_E(s) + \boldsymbol{J}\delta s + O\left(\|\delta s\|^2\right) \tag{1}$$

This linearization induces a linear-feedback-stabilized policy that at each time-step has a nominal action $a_t^\star$, but also expects to be in state $s_t^\star$, and correspondingly adjusts the nominal action with a linear correction based on discrepancy between the nominal and actual state at time t:

$$\mu_{FB}(s_t) = a_t^\star + \boldsymbol{J}_t^\star(s_t - s_t^\star), \quad \text{where} \quad \boldsymbol{J}_t^\star = \left.\frac{d\mu_E(s)}{ds}\right|_{s=s_t^\star} \tag{2}$$

We empirically validated that a linear feedback policy about the nominal trajectory of the expert can approximate the expert behavior reasonably well for clips we examine (see results Fig. 3).

To ensure that the student retains expert robustness, we would like expert actions $\mu_E(s)$ and student actions $\mu_\theta(s)$ to be close under a plausible (noisy) expert state distribution ρ_E. A surrogate loss used in imitation learning as well as knowledge transfer is the quadratic loss between actions (Ross et al., 2011) (or activations Srinivas & Fleuret (2018)).

$$\min_\theta \mathbb{E}_{s\sim\rho_E}[(\mu_E(s) - \mu_\theta(s))^2] \tag{3}$$

Behavioral cloning can refer to optimization of this objective, where ρ_E is replaced with an empirical distribution of a set of state-action pairs $\mathcal{S}$. This works well if $\mathcal{S}$ adequately covers the state distribution later experienced by the student. Anticipating and generating an appropriate set of states on which to train the student typically requires many rollouts and can thus be expensive. Above we presented the expert as a feedback controller operating in a tube around some nominal trajectory with states $s_1^\star, \ldots, s_T^\star$, actions $a_1^\star, \ldots, a_T^\star$, and Jacobians $\boldsymbol{J}_1^\star, \ldots, \boldsymbol{J}_T^\star$. We approximate ρ_E with the distribution of states introduced by state perturbations around this nominal trajectory:

$$\min_\theta \frac{1}{T}\sum_i \mathbb{E}_{\delta s_i\sim\Delta(s)}[\|\mu_E(s_i + \delta s_i) - \mu_\theta(s_i + \delta s_i)\|^2]. \tag{4}$$

However, this objective still requires expert evaluations at the perturbed states. Using the linearization described above we can replace the expert action $\mu_E(s + \delta s)$ with the Jacobian-based linear-feedback policy $\mu_{FB}(s + \delta s)$, which is available offline. This yields the LFPC objective:

$$\min_\theta \frac{1}{T}\sum_i \mathbb{E}_{\delta s_i\sim\Delta(s)}[||\mu_\theta(s_i^\star + \delta s_i) - a_i^\star + \boldsymbol{J}_i^\star\delta s_i||_2^2], \tag{5}$$

One potentially important choice is the perturbation distribution $\Delta(s)$. Ideally, we would like $\Delta(s)$ to be the state-dependent distribution induced by physically plausible transitions, but estimating this distribution may require potentially expensive rollouts which we are trying to avoid. A cheaper object to estimate is the *stationary* transition noise distribution induced by noisy actions, which can be efficiently approximated from a small number of trajectories. Empirically, we found the objective 5 to be relatively robust to some variations in Δ, and we use a fixed marginal distribution for all clips.

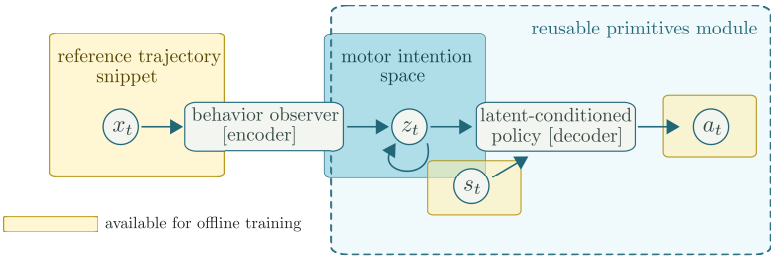


Figure 2: Neural probabilistic motor primitive architecture for one-shot skill deployment. Once the full model has been learned, the decoder-side can be reused in other settings.

Objective 5 bears interesting similarities to approaches such as denoising autoencoders (Vincent et al., 2008), where networks can learn to ignore local noise perturbations on inputs sampled from a high-dimensional noise distribution. Further, Mordatch et al. (2015) successfully distill feedback policies obtained from a planner. One question left open by this latter work is that of how much data might be required. Empirically we show in the experiments below that the augmented objective 5 can produce the desired robustness even from a very limited set of states.

There are multiple, relevant perspectives on LFPC. From one perspective, LFPC amounts to a data augmentation method. From another vantage, the approach attempts to match the mean action as well as the Jacobian at the set of points sampled along the nominal trajectory. Explicit Jacobian matching has been proposed elsewhere, for example in Czarnecki et al. (2017). See appendix B for further disambiguation relative to other approaches.

2.3 Neural probabilistic motor primitives

Our goal is to obtain a motor primitive module that can flexibly and robustly deploy, sequence, and interpolate a diverse set of skills from a large database of reference trajectories without any manual alignment or other processing of the raw experts. This requires a representation that does not just reliably encode all behavioral modes but also allows effective indexing of behaviors for recall. To ensure plausible and reliable transitions it is further desirable that the encoding of similar behaviors should be close in some sense in the representation space.

Compression of many expert skills via a latent variable model We achieve this goal by training an autoregressive latent variable model of the state-conditional action sequence which, at training time, is conditioned on short look-ahead snippets of the nominal/reference trajectory (see Fig. 2). As we demonstrate below this creates a sensory-motor representation space that allows the selective recall of particular behavioral modes via the conditional policy (decoder), and which also admits one-shot imitation via the trajectory encoder.

We use a model with one latent variable z_t per time step, modelling the state conditional action distribution. The encoder and decoder are distributions $q(z_t|z_{t-1}, x_t)$ and $p(a_t|z_t, s_t)$ where s_t is the state as in preceding sections and x_t is concatenation of a small number of future states $x_t = [s_t, ..., s_{t+K}]$. The encoder and decoder are MLPs with two and three layers, respectively. For architecture and experimental details see appendix D. The generative part of the model is given by:

$$p(a_{1:T}, z_{1:T}|s_{1:T}) = \prod_{t=1}^{T} p(z_t|z_{t-1})p(a_t|z_t, s_t) \tag{6}$$

Temporally nearby trajectory snippets should have a similar representation in the latent space. To encapsulate this intuition we choose an AR(1) process as a weak prior:

$$z_t = \alpha z_{t-1} + \sigma\epsilon, \;\; \epsilon \sim \mathcal{N}(0, I), \tag{7}$$

where $\sigma = \sqrt{1-\alpha^2}$, ensuring that marginally $z_t \sim \mathcal{N}(0, I)$, and set $\alpha = 0.95$ in most of our experiments. In future, it is interesting to investigate different values of α and learnable priors.

In order to train this model, we consider the evidence lower bound (ELBO):

$$\mathbb{E}_q \left[\sum_{t=1}^{T} \log p(a_t|s_t, z_t) + \beta\big(\log p_z(z_t|z_{t-1}) - \log q(z_t|z_{t-1}, x_t)\big) \right] \tag{8}$$

with a β parameter to tune the weight of the prior. For $\beta = 1$ this objective forms the well-known variational lower bound to $\log p(a_{1:T}|s_{1:T})$. This objective can be optimized using supervised learning (i.e. behavioral cloning from noisy rollouts) offline. Similarly, the objective can be adapted for use with linear-feedback policy cloning:

$$\mathbb{E}_{\delta s,q} \left[\sum_{t=1}^{T} \log p(a_t + \boldsymbol{J}_t \delta s_t | s_t + \delta s_t, z_t) + \beta\big(\log p(z_t|z_{t-1}) - \log q(z_t|z_{t-1}, x_t + \delta x_t)\big) \right] \tag{9}$$

where δs_t are i.i.d. perturbations drawn from suitable perturbation distribution Δ and δx_t is the concatenation of $[\delta s_t, \delta s_{t+1}, ..., \delta s_{t+K}]$.

Note we chose not to condition the encoder on actions, since we are interested in one-shot imitation in settings where actions are unobserved. In our experiments we experimented with different values of K and obtained similar performance. All the results reported in this paper use $K = 5$.[2]

Our architecture effectively implements a conditional information bottleneck between the desired future trajectory x_t and the action a_t given the past latent state z_{t-1} (similar to Alemi et al. (2017)). As discussed above the auto-correlated prior encourages an encoding in which temporally nearby latent states from the same trajectory tend to be close in the latent space, and the information bottleneck more generally encourages a limited dependence on x_t with z_t forming a compressed representation of the future trajectory as required for the action choice.

3 Experiments

3.1 Validation: transfer of single-behavior policies

To ground our results in a simple setting, we begin with transfer of a single-skill, time-indexed policy from one network to another. We compare the performance of various time-indexed policies for each of the experts depicted in Fig. 1. We compare the original expert policy, an open-loop action sequence along the experts nominal (i.e. mean) trajectory, a linear feedback policy along the expert nominal trajectory, as well as the network trained to match the linear-feedback behavior (LFPC). In addition we compare to policies trained from 100, 200, 500 or 1000 trajectories with behavioral cloning. We compare each approach with no action noise, small action noise, and moderate action noise (noise is i.i.d. normal per actuator with standard deviation magnitude .05 and .1 respectively, for action ranges normalized to $[-1, 1]$). Note that, open loop control almost always fails if the state is perturbed by even a small ϵ (though perhaps surprisingly, the backflip can almost be executed open loop due to limited ground contact). Remarkably, LFPC with a single trajectory performs on par with behavioral cloning based on hundreds of trajectories (see Fig. 3).

3.2 Core results: compressing thousands of experts

Having validated that single skills can be transferred, we next consider how well we can compress behaviors of the 2707 experts in our training set into the neural probabilistic motor primitive architecture. Assessing the models using the action-reconstruction loss is not very intuitive since it does not capture model behavior in the environment. Instead we report a more relevant measure based on expert imitation. Here we encode an expert trajectory into a sequence of latent variables and then *execute the policy in the environment* conditioned on this sequence. Note that this approach is open-loop with respect to the latents while being closed-loop with respect to state. We can then compare the performance of the trained system against experts on training and held-out clips according to the tracking reward used to train the experts originally. To account for different expert reward scales we report performance relative to the expert policy. Importantly, that this approach works is itself a partial validation of the premise of this work, insofar as open-loop execution of action sequences usually

[2]We also experimented with ways to look further into the future by conditioning on $x_t = [s_t, s_{t+1}, s_{t+3}, s_{t+6}, s_{t+10}, s_{t+15}]$. This gave broadly similar results.

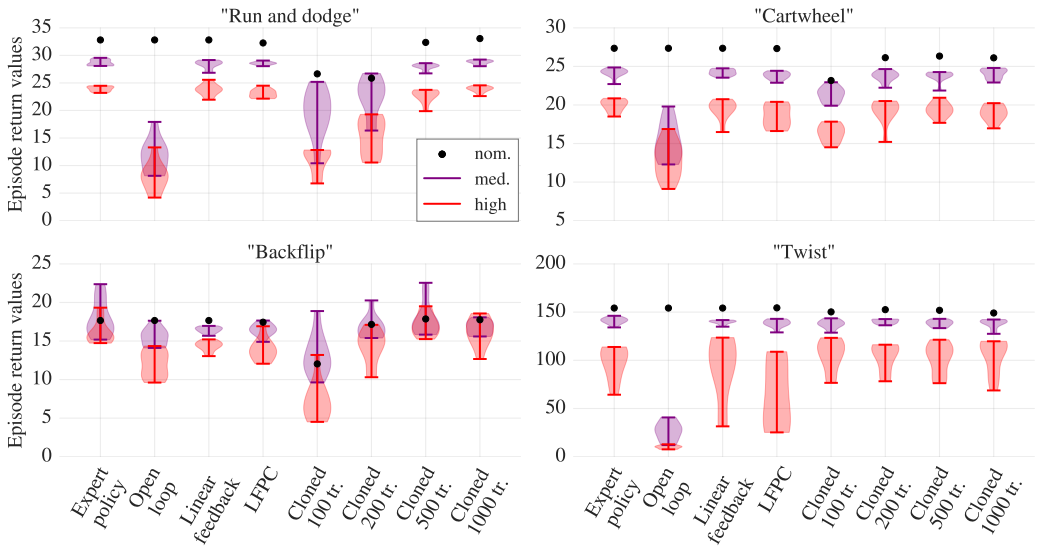


Figure 3: Comparisons of trajectory rollouts for 4 reference behaviors for the nominal trajectory and at varying noise levels. Note that the score is determined by similarity to motion-capture reference and the expert may be slightly suboptimal so slight improvements on the expert may arise by chance.

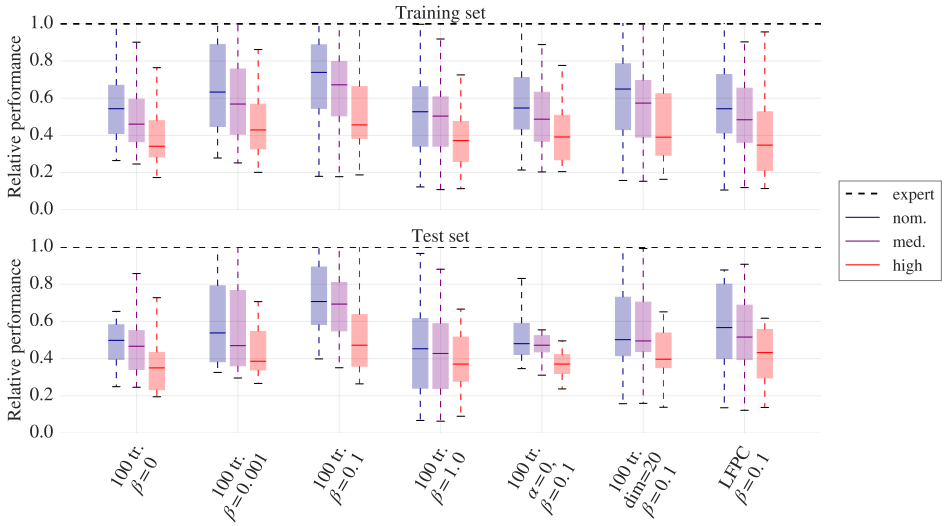


Figure 4: Performance relative to expert policies for trained neural probabilistic motor primitive models. Performance of model variations are compared on training and testing data. We compare models trained using cloning with 100 trajectories per expert for different levels of regularization, using a smaller latent space of dimension 20 rather than 60 in all other experiments, as well as LFPC.

trivially fails with minor perturbations. The trained neural probabilistic motor primitive module can execute behaviors conditioned on an open-loop noisy latent variable trajectory, implying that the decoder has learned to stabilize the body during latent-conditioned behavior.

There are a few key takeaways from the comparisons we have run (see Fig. 4). Most saliently cloning based on 100 trajectories from each expert with a medium regularization value ($\beta = 0.1$) works best. LFPC with comparable parameters works less well here, but has qualitatively fairly similar performance. Our ablations show that regularization and a large latent space are important

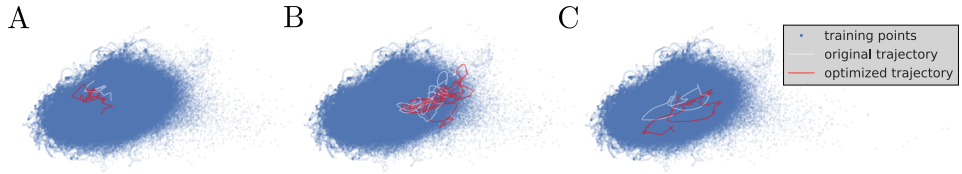


Figure 5: These panels consist of visualizations of the PCA latent space with comparisons in this space between one-shot latent-variable sequences and optimized latent variable sequences for various behaviors: A. Run B. Backwards walking C. Jumping. Running executes well based on the one-shot trajectory so serves as a reference for which optimization is not noticeably different. Walking backwards and jumping one-shot imitations fail, but are noticeably improved by optimization.

for good results. We also set the autoregressive parameter $\alpha = 0$ (instead of .95 in other runs), making the latent variables i.i.d.. This hurts performance, validating our choice of prior.[3]

3.3 Analysis of the trained model

When one-shot imitation of a trajectory fails, a natural question is whether the decoder is incapable of expressing the desired actions, or the encoder fails to encode the trajectory in such a way that the decoder will produce it. Let us be clear that we have no expectation that trajectories well outside the training distribution are likely to be either representable by the encoder or executable by the decoder. We propose an analysis to assess this for borderline cases. For held out trajectories that yield unsatisfying performance on one-shot imitation, we can simply optimize directly:

$$\min_{z_1 \dots z_T} \sum_{t=1}^{T} ||\mu_\theta(s_t, z_t) - a_t^\star||_2^2, \tag{10}$$

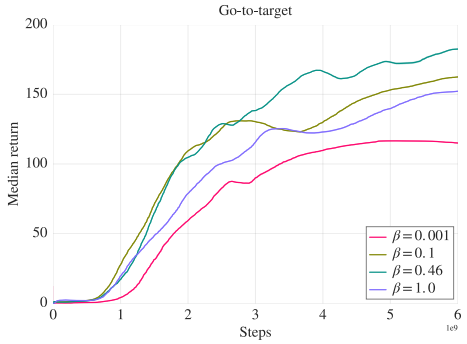
where μ_θ is the decoder mean. Empirically we see that this optimization meaningfully improves the executed behavior, and we visualize the shift in a three-dimensional space given by the first three principal components in Fig. 5.

We exhibit three examples where we visualize the original latent trajectory as well as the optimized latent trajectory. Performance is significantly improved (see supplementary video), showing the latent space can represent behaviors for which one-shot imitation fails. However execution remains imperfect suggesting that while much of the fault may lie with the encoder, the decoder still may be slightly undertrained on these relatively rare behavior categories. Quantitatively, among a larger set of clips with less than 50% relative expert performance for one-shot imitation we found that optimization as described above improved median relative expert performance from 43% to 78%.

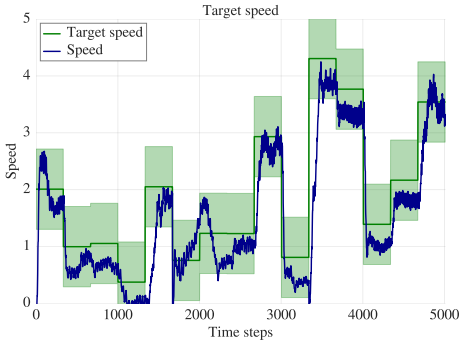
Other exploratory probes of the module suggest that it is possible in certain cases to obtain seamless transitioning between behaviors by concatenating latent-variable trajectories and running the policy conditioned on this sequence (e.g. in order to perform a sequence of turns). See additional supplementary video.

Reuse of motor primitive module Finally, we experimented with reuse of the decoder as a motor primitive module. We treat the latent space as a new custom action space and train a new high-level (HL) policy to operate in this space. At each time-step the high-level policy outputs a latent-variable z_t. The actual action is then given by the motor primitive module $p(a_t|s_t, z_t)$. A natural locomotion task that can challenge the motor module is a task which requires abrupt, frequently redirected movement with sharp turns and changes of speed. To implement this we provide the higher-level controller with a target that is constant until the humanoid is near it for a few timesteps at which point it randomly moves to another nearby location. While no single task will comprehensively probe the module, performing well in this task demands a wide range of quick locomotion behavior. For training we used SVG(0) (Heess et al., 2015) with the Retrace off-policy correction (Munos et al.,

[3]One other feature of the training pipeline we experimented with is mirroring of policies according to bilateral symmetry (thereby approximately doubling the expert data) – this improves results slightly and all models compared here use mirroring.



(a) Median return value across 10 seeds for the go-to-target task vs learner steps. Compared to a very weakly regularized module ($\beta = 0.001$), more regularized motor primitives modules both trained faster and achieved higher final performance.



(b) Our model is able to track the target speed accurately. Shown here are target speed and actual speed in the egocentric forward direction for three episodes. The reward function is a Gaussian centered at the target speed. The shaded region corresponds to $\pm$ one standard deviation.

Figure 6: Reuse of neural probabilistic motor primitive modules.

2016). With only a sparse task reward, the HL-controller can learn to control the body through the learned primitive space, and it produces rather humanlike task-directed movement. We observed that more regularized motor primitive modules had more stable initial behavior when connected to the untrained high-level controller (i.e. were less likely to fall at the beginning of training). Compared to a very weakly regularized module ($\beta = 0.001$), more regularized motor primitives modules both trained faster and achieved higher final performance (see Fig. 6a). We also investigated a go-to-target task with bumpy terrain that is unobserved by the agent. The fact that our model can learn to solve this task demonstrates its robustness to unseen perturbations for which the motor primitive module was not explicitly trained. In another experiment we investigated a task in which the agent has to move at a random, changing target speed. This requires transitions between qualitatively different locomotion behavior such as walking, jogging, and running (see Fig. 6b). See an extended video of these experiments.

We emphasize a few points about these results to impact their importance: (1) Using a pretrained neural probabilistic motor primitives module, new controllers can be trained effectively from scratch on sparse reward tasks, (2) the resulting movements are visually rather humanlike without additional constraints implying that the learned embedding space is well structured, and (3) the module enables fairly comprehensive and smooth coverage for the purposes of physics-based control.

4 Discussion

In this paper we have described approaches for transfer and compression of control policies. We have exhibited a motor primitive module that learns to represent and execute motor behaviors for control of a simulated humanoid body. Using either a variant of behavioral cloning or linear feedback policy cloning we can train the neural probabilistic motor primitive module capable of robust one-shot-imitation, and with the latter we can use relatively restricted data consisting of only single rollouts from each expert. While LFPC did not work quite as well in the full-scale model as cloning from noisy rollouts, we believe it holds promise insofar as it may be useful in rollout-limited settings, and there is room for further improvement as we did not carefully tune certain parameters, most saliently the marginal noise distribution Δ. In future work, we envision LFPC could be adapted for use in settings where rollouts are costly to obtain, such as real-world robots.

The resulting neural probabilistic motor primitive module is interpretable and reusable. We are optimistic that this kind of architecture could serve as a basis for further continual learning of motor skills. This work has been restricted to motor behaviors which do not involve interactions with objects and where a full set a of behaviors are available in advance. Meaningful extensions of this work may attempt to greatly enrich the space of behaviors or demonstrate how to perform continual learning and reuse of new skills.

ACKNOWLEDGMENTS

The data used in this project was obtained from mocap.cs.cmu.edu. The database was created with funding from NSF EIA-0196217.

REFERENCES

Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. Deep variational information bottleneck. *International Conference on Learning Representations*, 2017.

Anonymous. Hierarchical visuomotor control of humanoids. *Submission to International Conference on Learning Representations*, 2019.

Michael Athans. The role and use of the stochastic linear-quadratic-gaussian problem in control system design. *IEEE transactions on automatic control*, 16(6):529–552, 1971.

Trapit Bansal, Jakub Pachocki, Szymon Sidor, Ilya Sutskever, and Igor Mordatch. Emergent complexity via multi-agent competition. *International Conference on Learning Representations*, 2018.

E Bizzi, VCK Cheung, A d'Avella, P Saltiel, and Matthew Tresch. Combining modules for movement. *Brain research reviews*, 57(1):125–133, 2008.

Wojciech M Czarnecki, Simon Osindero, Max Jaderberg, Grzegorz Swirszcz, and Razvan Pascanu. Sobolev training for neural networks. In *Advances in Neural Information Processing Systems*, pp. 4281–4290, 2017.

Kai Ding, Libin Liu, Michiel Van de Panne, and KangKang Yin. Learning reduced-order feedback policies for motion skills. In *Proceedings of the 14th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 83–92. ACM, 2015.

Yan Duan, Marcin Andrychowicz, Bradly Stadie, OpenAI Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning. In *Advances in neural information processing systems*, pp. 1087–1098, 2017.

Tommaso Furlanello, Zachary C Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. *arXiv preprint arXiv:1805.04770*, 2018.

Michael Graziano. The organization of behavioral repertoire in motor cortex. *Annu. Rev. Neurosci.*, 29:105–134, 2006.

Nicolas Heess, Gregory Wayne, David Silver, Tim Lillicrap, Tom Erez, and Yuval Tassa. Learning continuous control policies by stochastic value gradients. In *Advances in Neural Information Processing Systems*, pp. 2944–2952, 2015.

Nicolas Heess, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, Ali Eslami, Martin Riedmiller, et al. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286*, 2017.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

David H. Jacobson and David Q. Mayne. *Differential Dynamic Programming*. Elsevier, 1970.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *International Conference on Learning Representations*, 2013.

Michael Laskey, Jonathan Lee, Roy Fox, Anca Dragan, and Ken Goldberg. Dart: Noise injection for robust imitation learning. In *Conference on Robot Learning*, pp. 143–156, 2017.

Libin Liu and Jessica Hodgins. Learning basketball dribbling skills using trajectory optimization and deep reinforcement learning. *ACM Transactions on Graphics (TOG)*, 37(4):142, 2018.

Libin Liu, KangKang Yin, Michiel van de Panne, Tianjia Shao, and Weiwei Xu. Sampling-based contact-rich motion control. In *ACM Transactions on Graphics (TOG)*, volume 29, pp. 128. ACM, 2010.

David Mayne. A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems. *International Journal of Control*, 3(1):85–95, 1966.

Josh Merel, Yuval Tassa, Sriram Srinivasan, Jay Lemmon, Ziyu Wang, Greg Wayne, and Nicolas Heess. Learning human behaviors from motion capture by adversarial imitation. *arXiv preprint arXiv:1707.02201*, 2017.

Igor Mordatch, Kendall Lowrey, Galen Andrew, Zoran Popovic, and Emanuel V Todorov. Interactive control of diverse complex characters with neural networks. In *Advances in Neural Information Processing Systems*, pp. 3132–3140, 2015.

Jun Morimoto and Christopher G Atkeson. Minimax differential dynamic programming: An application to robust biped walking. In *Advances in neural information processing systems*, pp. 1563–1570, 2003.

Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc Bellemare. Safe and efficient off-policy reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 1054–1062, 2016.

Gerhard Neumann, Christian Daniel, Alexandros Paraschos, Andras Kupcsik, and Jan Peters. Learning modular policies for robotics. *Frontiers in computational neuroscience*, 8:62, 2014.

Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. *arXiv preprint arXiv:1511.06342*, 2015.

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *arXiv preprint arXiv:1804.02717*, 2018.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pp. II–1278–II–1286. JMLR.org, 2014. URL `http://dl.acm.org/citation.cfm?id=3044805.3045035`.

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

Andrei A Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015.

Stefan Schaal, Jan Peters, Jun Nakanishi, and Auke Ijspeert. Learning movement primitives. In *International Symposium on Robotics Research, (ISRR)*, 2003.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pp. 1889–1897, 2015.

Suraj Srinivas and François Fleuret. Knowledge transfer with jacobian matching. *CoRR*, abs/1803.00443, 2018. URL `http://arxiv.org/abs/1803.00443`.

Yuval Tassa, Tom Erez, and Emanuel Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 4906–4913. IEEE, 2012.

Yuval Tassa, Nicolas Mansard, and Emo Todorov. Control-limited differential dynamic programming. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 1168–1175. IEEE, 2014.

Yee Whye Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 4499–4509, 2017.

Emanuel Todorov and Zoubin Ghahramani. Unsupervised learning of sensory-motor primitives. In *Engineering in Medicine and Biology Society, 2003. Proceedings of the 25th Annual International Conference of the IEEE*, volume 2, pp. 1750–1753. IEEE, 2003.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pp. 1096–1103. ACM, 2008.

Ziyu Wang, Josh S Merel, Scott E Reed, Nando de Freitas, Gregory Wayne, and Nicolas Heess. Robust imitation of diverse behaviors. In *Advances in Neural Information Processing Systems*, pp. 5320–5329, 2017.

APPENDICES

A MOTION CAPTURE EXPERTS

The approach we use for producing experts is detailed more fully in Anonymous (2019). In short, this approach for producing experts largely follows Peng et al. (2018). We took the energy function proposed in SAMCON (Liu et al., 2010), and use it as a per timestep reward to train a time-indexed policy that tracks/imitates a motion capture reference clip (Peng et al., 2018). As proposed in Merel et al. (2017); Peng et al. (2018), episodes are initialized to poses throughout the motion capture reference and episodes are early-terminated when the character falls. Here we use an off-policy RL algorithm, SVG(0) (Heess et al., 2015) with Retrace (Munos et al., 2016). As done in Merel et al. (2017); Peng et al. (2018) and elsewhere, we train stochastic policies and use the mean (i.e. noiseless) action as the expert policy.

B RELATIONSHIP TO OTHER KNOWLEDGE TRANSFER IDEAS

Firstly, we note that the emphasis of the proposal in this work is to match the responsivity of the expert policy in a neighborhood around each state. This is distinct from activation matching or KL matching where the emphasis is on matching the action/activation distribution for a particular state (Rusu et al., 2015; Teh et al., 2017). Secondly, we emphasize that the kind of robust knowledge transfer we discuss here is distinct from that which is seen to be important in other settings. For example Srinivas & Fleuret (2018) provide a line of reasoning that involves training a student system to match the exact activations of a teacher in the presence of perturbations on the student inputs. This logic is sound in the setting of large-scale vision systems. However in the context of control policies, this would look like:

$$\min_{\theta} \sum_{s \in \mathcal{S}^\star} \mathbb{E}_{\delta s \sim \Delta(s)}[(\mu_E(s) - \mu_\theta(s + \delta s))^2] \tag{11}$$

This essentially means that the student policy is learning to "blindly" reproduce the action of the expert exactly, despite input perturbations. While this is well motivated if the noise is thought to be orthogonal to the proper functioning of the system, this is a very bad idea for control, where you need to pay close attention to small input perturbations. Technically, this amounts to setting the local feedback to zero, and behaving in a sort of *open-loop*-like fashion.

C VISUALIZATION OF STATIONARY POLICY BEHAVIOR

Locomotion behavior is, at least in the simplest case roughly a limit cycle. In an additional experiment to test LFPC we gathered three gait cycles of running behavior and performed LFPC. Note that here the student policy need not be time-indexed even when the demonstrations were time-indexed. This restricted case shows striking generalization in the presence of noise (see Fig. A.1).

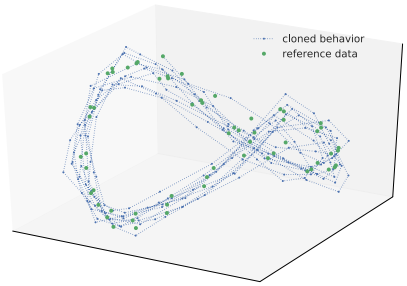


Figure A.1: Dimensionality reduction (PCA) performed on set of poses obtained from noisy rollouts of the stationary cloned policy (blue). The limited reference data originating from a time-indexed policy has been projected into the same space (green). Observe that the rollouts are considerably noisier and consistently deviate from the reference trajectory, nevertheless the cloned-policy trajectories return to the limit cycle.

D ARCHITECTURE AND TRAINING DETAILS

The decoder $p(a_t|s_t, z_t)$ in our experiments was a MLP with three layers with 1024 hidden units taking as input the concatenation of state s_t and latent variable z_t. The decoder output distribution is a multivariate Gaussian with fixed standard deviation of 0.1 (action values are normalized to $[-1, 1]$). We found that fixing the standard deviation made it significantly easier to prevent overfitting. Note that in this setting varying the β parameter is equivalent to varying the fixed output variance (up to a constant). The encoder $q(z_t|z_{t-1}, x_t)$ in our experiments was also an MLP with two layers of 1024 hidden units each. The inputs were simply concatenated at the input. The encoder output distribution was a multivariate Gaussian with learnt variance. In most of our experiments, we used a 60-dimensional latent space.

We used the reparametrization trick (Kingma & Welling, 2013; Rezende et al., 2014) to train the model and used stochastic gradient descent with ADAM (Kingma & Ba, 2015) with a learning rate of 0.0001. In the case of models trained on 100 trajectories per expert we used minibatches of 512 subsequences of length 30.

For LFPC we sampled 32 subsequences of length 30 and produced 5 perturbed state sequences per subsequence. In preliminary experiments the length of the subsequences did not have a major impact on model performance.