Imitation Learning of Factored Multi-agent Reactive Models

Michael Teng
Engineering Science
University of Oxford
mteng@robots.ox.ac.uk

Tuan Anh Le
Engineering Science
University of Oxford
tuananh@robots.ox.ac.uk

Adam Scibior
Computer Science
University of British Columbia
ascibior@cs.ubc.ca

Frank Wood
Computer Science
University of British Columbia
fwood@cs.ubc.ca

Abstract

We apply recent advances in deep generative modeling to the task of imitation learning from biological agents. Specifically, we apply variations of the variational recurrent neural network model to a multi-agent setting where we learn policies of individual uncoordinated agents acting based on their perceptual inputs and their hidden belief state. We learn stochastic policies for these agents directly from observational data, without constructing a reward function. An inference network learned jointly with the policy allows for efficient inference over the agent's belief state given a sequence of its current perceptual inputs and the prior actions it performed, which lets us extrapolate observed sequences of behavior into the future while maintaining uncertainty estimates over future trajectories. We test our approach on a dataset of flies interacting in a 2D environment, where we demonstrate better predictive performance than existing approaches which learn deterministic policies with recurrent neural networks. We further show that the uncertainty estimates over future trajectories we obtain are well calibrated, which makes them useful for a variety of downstream processing tasks.

1 Introduction

[1] Imitation learning is the task of learning to simulate the behavior of agents whose actions we observe in a given environment (Schaal, 1999). The settings can be very diverse, such as tracking animal movement patterns, holding a dialogue with another agent, or emulating human players in video games. We focus on situations where we have a good model of the environment and the task is to learn a policy governing the agent's actions, which can be alternatively seen as a generative model of the observed actions. Learning good generative models for the behavior of real world agents is of direct interest, such as for constructing realistic simulators (Abbeel and Ng, 2004), as well as a useful auxiliary for solving other problems, for example as a means of incorporating expert demonstration into reinforcement learning systems (Osa et al., 2018).

The classic approach to imitation learning is to learn a predictive model for the agent's actions in a supervised fashion. A simple example of this approach would be to train a recurrent neural network (RNN) to predict the agent's actions or distributions over actions (Wen et al., 2015). While this can work well in certain cases, learning deterministic policies is not ideal, since it is generally recognized that biological agents behave stochastically, so using a deterministic model places substantial limits on how faithfully the agents' behavior can be recovered (Bayley and Cremer, 2001; Kaelbling, 1993). While it is possible to modify the predictive model to produce a probability distribution over actions, this approach requires picking a parameterization over actions, which is inconvenient, especially when the actions come from a continuous set. In the context of RNNs, the evolution of the latent state is constrained to be deterministic conditionally on inputs, which is arguably also an unrealistic assumption.

In recent years there has been an explosion of interest in deep generative models, which use neural networks transforming random inputs to learn complex probability distributions. In particular variational auto-encoders (VAEs) (Kingma and Welling, 2013), and structured variants thereof, can be used to simultaneously learn a generative model and an associated inference network. A particular variant of this technique, called variational recurrent neural network (VRNN) (Chung et al., 2015) is suitable for modeling sequential data. In this

[1]Under review at Uncertainty in Artificial Intelligence 2019

work we use VRNNs to perform imitation learning, obtaining stochastic policies parameterized by neural networks. This model choice naturally produces stochastic policies and it allows the latent state to evolve in a stochastic fashion.

The application of deep generative models to imitation learning is itself not novel. However, unlike previous work that focused on modeling agents from an external perspective (Johnson et al., 2016; Zhan et al., 2019), we learn a model where an agent acts based on their perceptual input only, which is arguably a more realistic assumption. The contribution of this work is to show that deep generative models are suitable tools for imitation learning of biological agents in this egocentric setup and to demonstrate that they outperform approaches based on RNNs (Eyjolfsdottir et al., 2016). We not only learn stochastic policies that generate more realistic agent behavior, but also demonstrate that following these policies produce a calibrated uncertainty estimate for the future states of the agent over time. These uncertainty estimates can be useful for a variety of downstream tasks, such as calculating confidence regions over future joint positions of moving agents.

Our work targets pure imitation learning, treating it as a goal in and of itself. Thus performance is judged in terms of how well the learned model resembled the behavior of real agents, using a variety of concrete metrics. This is in contrast to reinforcement learning, where the goal is to find a policy that solves a particular task well (Sutton and Barto, 2018). A closely related approach is that of inverse reinforcement learning (Sermanet et al., 2016), where the goal is to learn a reward function from the agent's actions, assuming that the actions are near optimal. Such a learned reward function can then be used to find a policy that achieves the same goals through different means. Since in this work we are not interested in learning policies other than the one actually executed by the agents, we do not pursue this direction and do not attempt to learn or utilize any reward functions. Instead, we directly learn the policy from the agent's actions.

We work in a multi-agent environment, but assume no explicit communication or coordination between the agents. Each agent executes their own policy and only learns about other agents' behavior through its perceptual inputs. In the experiments we assume that all the agents execute the same policy for simplicity, but our approach is straightforward to extend to settings with multiple agents with different policies or different action spaces since we can learn all the policies independently. This is in contrast to work such as (Zhan et al., 2019), which uses explicit coordination variables shared between different agents.

As a concrete application we learn to imitate the behavior of fruit flies (*Drosophila melanogaster*) in a 2D environment. This setup is adapted from Eyjolfsdottir et al. (2016), who perform the same imitation learning task using RNNs. We compare with their results, demonstrating more realistic behavior and showing that we obtain better-calibrated uncertainty estimates for future fly locations.

In the rest of the paper, Section 2 reviews the relevant background information, Section 3 describes our model, including its particular realization for the fly modeling task, and Section 4 presents the experimental results.

2 Background and related work

2.1 Variational and importance-weighted autoencoders

Variational autoencoders (VAEs) (Kingma and Welling, 2013) is an approach to deep generative modeling where we simultaneously learn a generative model and an amortized inference network. The latter is crucial for performing inference efficiently, which is normally very difficult in deep generative models. Once trained, the inference network, given a dataset, produces parameters of a variational approximation for the posterior conditional on this dataset without any further optimization. The generative model is parameterized by θ and consists of a latent variable z and data x, denoted as $p_\theta(z, x) = p_\theta(z)p_\theta(x|z)$. The inference network only targets the conditional distribution written as $q_\phi(z|x)$ and is parameterized by ϕ. The model parameters θ and inference network parameters ϕ are typically neural network weights which need to be learned.

In order to train these networks, Kingma and Welling (2013) propose maximizing the evidence lower bound (ELBO), which is a sum over ELBOs for individual data points, defined as:

$$\mathrm{ELBO}(\theta, \phi, x) \tag{1}$$

$$:= \log p_\theta(x) - \mathrm{KL}(q_\phi(z|x) \parallel p_\theta(z|x)) \tag{2}$$

$$= \mathbb{E}_{q_\phi(z|x)}\left[\log \frac{p_\theta(z, x)}{q_\phi(z|x)}\right]. \tag{3}$$

The ELBO is maximized jointly over θ and ϕ. Equation 2 justifies using the ELBO as an optimization target for θ, since it lower bounds the marginal likelihood and therefore can be seen as an approximation to Bayesian model selection, and for ϕ, since it is proportional to the

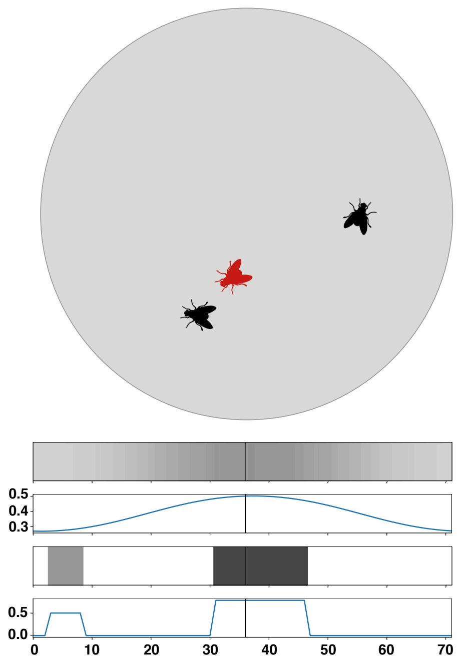


Figure 1: Diagram of the red fly's field of view encoding in a petri dish environment containing two other flies. The middle line in the plots denotes the fly's direct line of sight. Each pair of plots indicates the agent's field of view with respect to walls and other flies, respectively. The fly closer to the red agent contributes more mass to the encoding vector.

Kullback-Leibler divergence measuring the distance between the variational approximation and the true posterior.

Equation 3 gives a formula for calculating the ELBO. Taking gradients of this expression with respect to θ and ϕ and approximating the expectation by a single sample from q, we obtain formulae for stochastic gradients used in the optimization process. The stochastic gradient with respect to θ usually does not pose a serious problem, but the one with respect to ϕ involves a score function term, resulting from taking a gradient with respect to the parameters of the distribution over which the expectation is taken, which can have too high variance. For certain classes of continuous distributions this problem can be ameliorated by using the reparameterization trick, but for discrete variables that is not possible.

Burda et al. (2015) propose an extension to the VAE where, for a given number of particles K, the single-data ELBO is instead defined as:

$$\mathrm{ELBO}^K_{\mathrm{IWAE}}(\theta, \phi, x) \tag{4}$$

$$= \mathbb{E}_{z_1,\cdots,z_K \sim q_\phi(z|x)} \left[\log \left(\frac{1}{K} \sum_{k=1}^{K} \frac{p_\theta(z_k, x)}{q_\phi(z_k|x)} \right) \right]. \tag{5}$$

For $K = 1$ this is equivalent to the standard ELBO. Increasing K leads to a tighter lower bound on $p_\theta(x)$, which translates into improvement in the learned model. Howerver, increasing K, perhaps surprisingly, may result in a worse inference network (Rainforth et al., 2018).

In this paper we use the approach advocated by Le et al. (2018), who adapt the reweighted wake-sleep algorithm (Bornschein and Bengio, 2014) to the context of learning deep generative models. The approach is to use distinct objectives for learning θ and ϕ. For θ we use the IWAE objective from Equation 5. For ϕ, however, we derive another objective targeting the KL in the opposite direction. This is justified by regarding q_ϕ as a proposal distribution for importance sampling rather than a variational distribution. The target is as follows.

$$E_{p(x)} \left[-KL(p_\theta(z|x) || q_\phi(z|x)) \right] \tag{6}$$

$$= E_{p(x)} \left[E_{p_\theta(x)} \left[\log(q_\phi(z|x)) - \log(p_\theta(z|x)) \right] \right] \tag{7}$$

Since this formula does not involve any expectations with respect to q_ϕ, taking the gradient with respect to ϕ does not produce any score function terms and therefore has reasonably low variance, even in the presence of discrete variables. This is crucial for us, since our model includes discrete variables. We note, however, that there exist variance reductions methods that can be used with standard VAE and IWAE objectives in the presence of discrete variables, based on continuous relaxation (Maddison et al., 2016; Jang et al., 2016) or control-variate methods (Mnih and Rezende, 2016; Mnih and Gregor, 2014; Tucker et al., 2017; Grathwohl et al., 2017).

2.2 Variational Recurrent Neural Network

The variational RNN (Chung et al., 2015) is a deep latent generative model that is an extension in the VAE family. It can be viewed as an instantiation of a VAE at each timestep, with the model factorised in time overall. We show a particular variant of VRNN we use in this work as a graphical model in Figure 2. Similar models have been applied to imitation learning of mouse behavior Johnson et al. (2016) and the tactics of basketball players Zhan et al. (2019). Those authors used a god's eye view of the entire environment as inputs, where we instead use perceptual inputs available to individual agents, thus learning in an ego-centric setting.

2.3 Fly behavior dataset

The Fly-vs-Fly dataset we use (Eyjolfsdottir et al., 2014) contains annotated tracks of fruit flies interacting with each other. In order to expose this to our general model, we are interested in the most basic representation or encoding of perceptual input and behavioral actions. At each timestep, the fly has a field of view available to it, which can contain solid surfaces (walls of the petri dish) and any number of other flies. In keeping with Eyjolfsdottir et al. (2016), the agent's visual field is divided into 72 individual slices, and the first index encodes the inverse distance to an object starting at the slice directly behind the fly's orientation, (i.e. 180 degrees). This procedure is repeated for each slice going clockwise, until slice 72 is again at 180 degrees. This provides two generic visual field encodings for the fly, one denoting walls and the other denoting flies (see Figure **??**). Finally, this encoding scheme takes care of realistic conditions such as occlusion, multiple other flies, and new environments as well.

Next, the action space is treated as follows: for each fly, its permissible actions are forward and backward motion, changing its wing angle, changing its wing length, extending and contracting its body (thereby producing a change in the visual field of other agents around it), and finally yawing or turning in place. At each timestep, these actions are encoded by a delta to the previous position. That is, the fly knows where it currently is and chooses for each of the 9 discrete actions, some delta away from its position in the corresponding unit of measurement. If a fly wishes to walk towards an object at its 3 o'clock, it will produce a 90 degree turn, followed by a movement forward some number of units. Another example is during a mating ceremony, male flies often encircle the female and vigorously flap its wings, which is represented by a series of sharp and quick wing deltas and changing of angles.

Within the dataset, tracking data for these features are available as absolute position taken at a resolution of 30 frames per second. Using the absolute positional data, deltas are calculated with respect to the aforementioned action space.

Eyjolfsdottir et al. (2016) learn a generative model for fly behavior, but use only the deterministic ladder RNN that at each step generates parameters for a distribution from which an action is sampled. Moreover, they are concerned with unsupervised detection of behavior. We instead use a VRNN model, which is arguably more biologically plausible in allowing stochastic transitions within the agent's latent state. Our approach generates more realistic behaviors and provides better calibrated uncertainty estimates.

3 Model

In this section we describe our generative model and the associated inference networks. We simultaneously describe the general architecture and its particular realization for the fly tracking model. While the model itself is very general, having this concrete example helps to provide more intuition about the roles of different components.

For notational purposes, we denote the discrete timestep as t, and the individual agent in question as f. Our model is factorized in agents, that is every agent executes the same policy and the agents only interact with each other through perceptual observations. The model builds off of the VRNN, where transitions and memory are embedded into the recurrence of a deterministic RNN.

3.1 Notation

We first establish notation for the model. During this, we explain the decisions for this encoding scheme in the context of biological plausibility.

Discrete Mental State Let $a_{t,f}$ denote a discrete latent variable, which we intuit as a discrete high-level state of cognition being activated during the agent's planning.

Continuous Mental State Let $c_{t,f}$ denote the continuous cognitive state, which allows the model to have more capacity to represent the internal state of the agent.

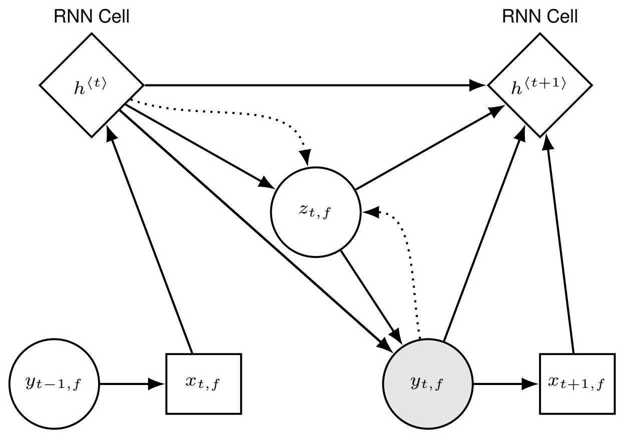
The random variables, $a_{t,f}$ and $c_{t,f}$, together comprise the latent random variable used by the agent f at time t. We write $z_{t,f} = \{a_{t,f}, c_{t,f}\}$ to denote this.

Spatial Localization We write $y_{t,f}$ to denote the action taken by agent f at time t. In the fly model, the action is a 9-dimension vector, whose elements are the respective deltas of a permitted action from the perspective of the fly. Let:

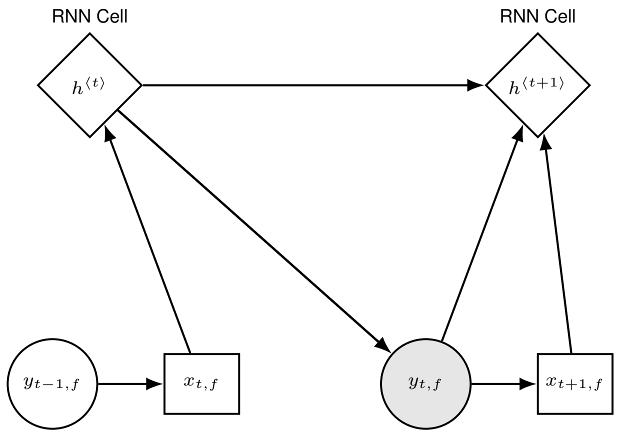
$$y_{t,f} = \{o_{t,f}, m_{t,f}^{fwd}, m_{t,f}^{lat}, w_{t,f}^{ll}, w_{t,f}^{la}, w_{t,f}^{rl}, w_{t,f}^{la}, b_{t,f}^{maj}, b_{t,f}^{min}\}$$

For simplicity, all of the following are understood as the deltas, or the change in the specified variable at a given timestep:

- Let $o_{t,f}$ denote the orientation of the fly.
- Let $m_{t,f}^{fwd}$ and $m_{t,f}^{lat}$ denote the motion parallel to and orthogonal to the fly's orientation, respectively (i.e. forward and lateral movement)



(a) VRNN-PI



(b) RNN

Figure 2: Graphical models comparing our proposed VRNN-PI (a) with the baseline RNN (b). The VRNN-PI can be thought of as a VAE at a time, t, parameterized by the current set of attribution and positional data, $x_{t,f}$ and the hidden units of the GRU cells. Dotted lines indicate inference, and solid lines the generative model. The RNN has no latent random variables and as a consequence no inference network.

- Let $\{w^{ll}_{t,f}, w^{la}_{t,f}, w^{rl}_{t,f}, w^{la}_{t,f}\}$ denote the left wing length, left wing angle, right wing length, and right wing angle, respectively. Wing angles are measured with respect to the axis given by the fly's current orientation.
- Let $b^{maj}_{t,f}$ and $b^{min}_{t,f}$ denote the fly body major and minor axis length. While the flies do not actually change their body size, they might reorient themselves in the third dimension, for example by climbing the walls of the dish, which in 2D view results in changing their body size.

For clarity, at each timestep, the observed motions $m^{fwd}_{t,f}$ and $m^{lat}_{t,f}$ are measured with respect to the fly's new orientation, after it makes a rotation in place according to $o_{t,f}$. For these actions, each can be thought of as a velocity of sorts, with the basis vector being the fly's own body axis. Cueva and Wei (2018) found that modeling movement using velocities leads to the emergence of neurological grid cells resemblance in the RNN parametrization, which provides a rationale for this encoding.

Sensory Encoding Let $x_{t,f}$ denote the sensory input of agent f at time t. In the graphical model terminology we consider $x_{t,f}$ to be a single node of a constraint network, i.e. a random variable whose value is known given the values of all other nodes.

In the fly model this consists of the fly's visual input and the relative positions of its body parts

- Let $s^{wall}_{t.f}$ denote 72-dimensional visual input of surrounding walls. Each slice contains the inverse Euclidean distance to an object in the field of view, with 0 denoting no object present.
- Let $s^{fly}_{t,f}$ denote the 72-dimensional visual input of other agents/flies present, with the same formula as above.
- Let $\{\hat{o}_{t,f}, \hat{w}^{ll}_{t,f}, \hat{w}^{la}_{t,f}, \hat{w}^{rl}_{t,f}, \hat{w}^{la}_{t,f}, \hat{b}^{maj}_{t,f}, \hat{b}^{min}_{t,f}\}$ encode the flies current physical state, which are body and wing configurations. Note that unlike the actions, these are specified as absolute values and not deltas. We include knowledge of the fly's global orientation, since flies are known to have internal compasses (Clandinin and Giocomo, 2015).

Together these values constitute the fly's perceptual input. Note that the fly does not have direct perception of its position in space, but can infer that information from the distances to walls in different directions.

$$x_{t,f} = \{s^{wall}_{t.f}, s^{fly}_{t,f}, \hat{o}_{t,f}, \hat{w}^{ll}_{t,f}, \hat{w}^{la}_{t,f}, \hat{w}^{rl}_{t,f}, \hat{w}^{la}_{t,f}, \hat{b}^{maj}_{t,f}, \hat{b}^{min}_{t,f}\}$$

3.2 Generation and Inference

Let c_f denote the initial state of the agent f and $p(c_f)$ be the prior for it. In the fly model, c_f is the initial coordinates and body position of the fly f and the prior is a uniform distribution over the permitted values of c_f.

At each time step the sensory inputs $x_{t,f}$ are calculated as a deterministic function ζ of the initial conditions and the actions of all the agents up to this point. In practice this is implemented by maintaining a representation of the complete state of the world, unavailable to any agent,

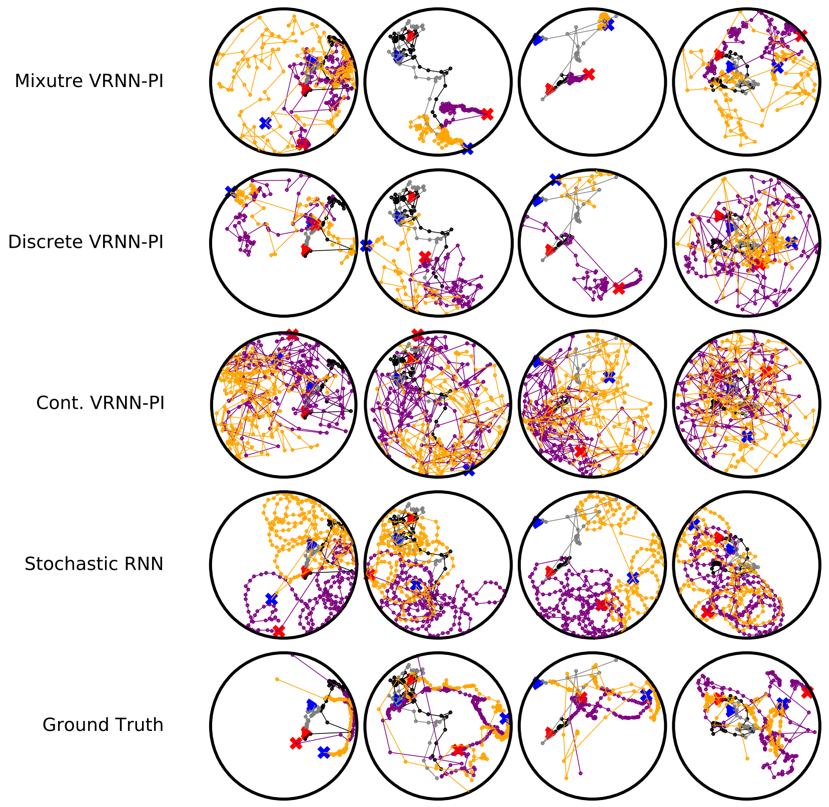


Figure 3: Comparison of each model's continuation tracks for four distinct seed sequences. Each row shows a model's continuation sampled from the generative models compared against the bottom row ground truth continuation. Each column shows one of four seed sequences used to seed the generative models. Within the petri dish, both flies' locomotion tracks are shown. For the first fly, the red arrow indicates the starting position of the seed sequence, the black markers and line indicate the seed tracks, the purple markers and line indicate the sampled continuation tracks, and the red 'x' indicates the final position of the fly after 200 timesteps. For the second fly, these indicators are blue arrow, gray markers, orange markers, and blue 'x', respectively. Qualitatively, the VRNN-PI variants using fully discrete or a mixed latent space are more realistic and exhibit identifiable behavior, such as slowly exploring and zipping around. The continuous VRNN-PI variant is highly erratic with large step sizes at almost every timestep and the stochastic RNN baseline exhibits mostly unrealistic constant step size and highly regular circular locomotion.

updating it using agents' actions at each step, and generating perceptual inputs for individual agents from this representation.

We then use $x_{t,f}$ to update the latent state $h_{t,f}$, sample latent variables $z_{t,f}$, and sample the action $y_{t,f}$.

- For $0 < t < T$, for f in $\{1, \dots, F\}$:

$$\begin{aligned} x_{t,f} &= \zeta(c_{1:F}, y_{1:t-1,1:F}) \\ h^{\langle t,f \rangle} &= \gamma_\psi\left(h^{\langle t-1,f \rangle}, z_{t-1,f}, x_{t,f}, y_{t-1,f}\right) \\ z_{t,f} &\sim p_{\theta_1}(\cdot | h^{\langle t,f \rangle}) \\ y_{t,f} &\sim p_{\theta_2}(\cdot | h^{\langle t,f \rangle}, z_{t,f}) \end{aligned}$$

The joint probability of the above model factorizes as:

$$p(z_{1:T,1:F}) = \prod_{f=1}^{F} \prod_{t=1}^{T} p_{\theta_1}(z_{t,f} | h^{\langle t,f \rangle}) p_{\theta_2}(y_{t,f} | h^{\langle t,f \rangle}, z_{t,f}) \tag{8}$$

The proposal distribution is as follows:

$$q_\phi(z_{1:T,1:F}) = \prod_{f=1}^{F} \prod_{t=1}^{T} q_{\phi_1}(z_{t,f} | h^{\langle t,f \rangle}, x_{t,f}) \tag{9}$$

The novel aspect of this model is that it feeds perceptual inputs for the agent into the VRNN. Thus we refer to this model as VRNN with perceptual inputs (VRNN-PI). It is depicted graphically in Figure 2a.

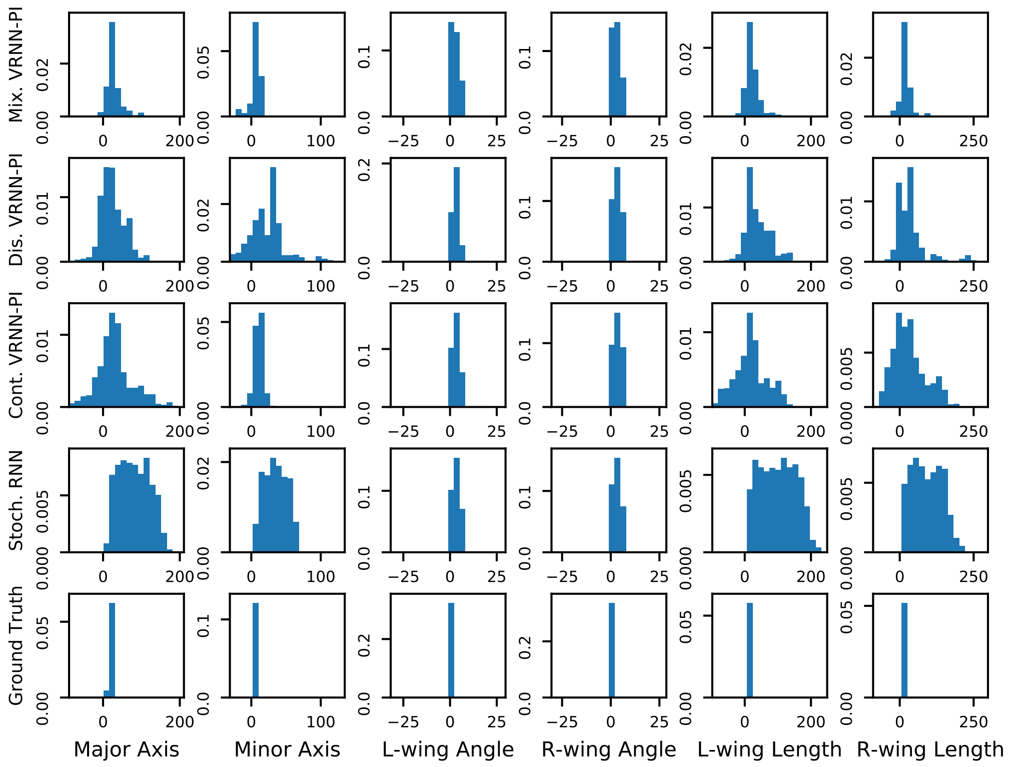


Figure 4: Comparison of the empirical distribution of $\{\hat{w}^{ll}_{t,f}, \hat{w}^{la}_{t,f}, \hat{w}^{rl}_{t,f}, \hat{w}^{la}_{t,f}, \hat{b}^{maj}_{t,f}, \hat{b}^{min}_{t,f}\}$. Each feature (by column) is averaged over time, fly, and eight distinctly seeded, 200 timestep sample continuations for each model (by row), while the ground truth is averaged over the training dataset. Each generative model histogram contains 3200 measurements while the ground truth contains 1,316,800 measurements. The means of the distributions for each VRNN-PI variant match closely with the ground truth but with greater noise, while the stochastic RNN feature distributions are highly noisy and farther from the ground truth mean than the VRNN-PI models.

4 Experiments

Here we report experimental results for our models. We use the male-to-male interactions of the Fly-vs-Fly dataset to get annotated tracks of 2 males interacting in a petri dish of diameter 130. The data is first cleaned into sequences of 160-dimensional vectors representing, $\{y_{t,f}, x_{t,f}\}_{t=1}^{T=200}$ for each fly. During training, we always balance the number of training examples from fly-1 and fly-2 in a minibatch, but this is unnecessary given our model (which is factorized in flies). The assumption here is that behavior modes are invariant to an individual fly being selected, if all the attributes are encoded similarly at a high level.

As a baseline, we implement the hierarchal RNN detailed in Eyjolfsdottir et al. (2016). Please refer to their paper for more details. At a high level, they use two parallel RNNs with diagonal connections between the first RNN outputs and the second. The first RNN takes $x_{t,f}$ and $y_{t,f}$ as input and the second RNN decodes its hidden state into the predicted actions. We note here one advantage of the deterministic RNN is its speed to train compared to our model. In order to obtain a stochastic policy, Eyjolfsdottir et al. (2016) discretize the actions and use the RNN to produce a probability vector over thus obtained discrete space, which is subsequently used to sample an action. With VRNN-PI we do not need any discretization.

Apart from comparing with the RNN baseline, we investigate the use of discrete and continuous latent variables in the VRNN-PI model. Specifically, in all experiments we use three variants of the model: one with only discrete latents (Discrete VRNN-PI), one with only continuous latents (Continuous VRNN-PI), and one with a mixture of discrete and continuous latents (Mixture VRNN-

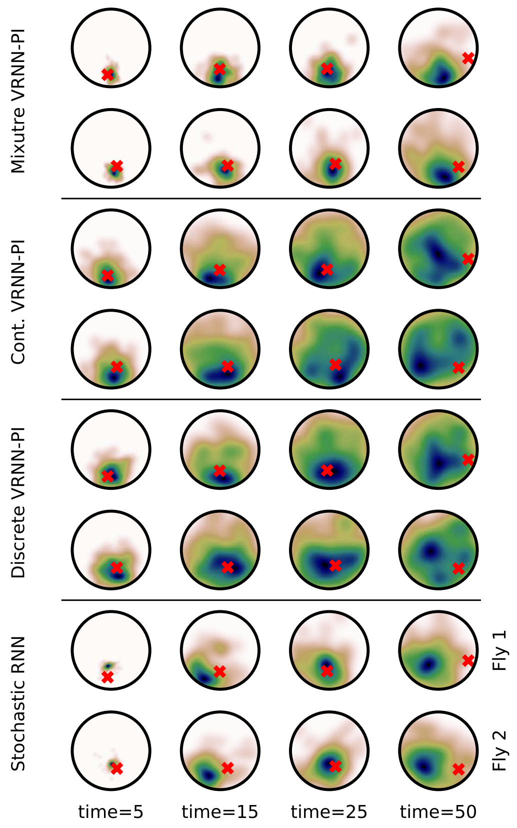


Figure 5: Comparison of uncertainty estimates over future trajectories of two flies' interactions in the environment between VRNN-PI models and the stochastic RNN. Each model is seeded with the same ground truth sequence each time and 100 continuations are sampled for 50 timesteps into the future. For columns 2-5, we show the kernel density estimate of all fly positions at the indicated timestep across the 100 continuations. Each model's estimate is split by fly for clarity (two rows per model indicated by separators), where the red 'x' indicates the true position of the fly at that timestep. The future trajectories across all models show greater uncertainty about the fly's position as the model evolves in time. The mixture VRNN-PI model is the least noisy as it evolves, and still captures the actual position within a high probability region, excluding fly 1 at time 50.

PI). We use a 120-dimensional Gaussian for the continuous variant, 60 one-hot encoded binary variables in the discrete variant, and a 60-dimensional Gaussian with 30 one-hot encoded binary variables for the respective latent codes in order to maintain a consistent neural architecture across variants. In all experiments we find that the mixture variant performs significantly better than the other two. See Section 5 for the discussion of these results.

Our first experiment is to condition the model on an initial sequence of actions, then sample a continuation and visually inspect how it compares with the true continuation that was not shown to the model. The results are shown in Figure 3. We find that under the RNN model the flies tend to move in circular patterns with relatively constant velocity. In contrast, real flies tend to alternate between fast and slow moves, changing their directions much more abruptly. All three variants of our VRNN-PI model recover this behavior, however Continuous VRNN-PI tends to behave too erratically. We were not able to identify any other clear visual artifacts in the generated trajectories that disagrees with the real data.

In the next experiment we compare various statistics of the flies' state over the course of time between the dataset and the simulations from different models. Figure 4 shows histograms of flies' positions. We see that in the real dataset those characteristics are very peaked, while in the models they are much more spread out, which we attribute to the quality of predictions deteriorating over time. We also note that the ground truth distributions are averaged over the entire training dataset, which contains

model	major axis	minor axis	l-wing angle	r-wing angle	l-wing length	r-wing length
Mixed	**1.42779**	**1.60993**	**1.56187**	**1.60134**	**1.52634**	**1.58506**
Discrete	1.76864	1.87270	1.79445	1.86377	1.64156	1.84672
Gauss	1.66578	1.69824	1.79259	1.82584	1.68953	1.78645
RNN	1.86795	1.92653	1.93097	1.95274	1.82537	1.84103

Table 1: Comparison of histogram results between models and their respective feature distributions. Equal bin sizes of 1 were used across all features and models and distance is computed using L1-norm between the model results and the ground truth. The closest distances are highlighted in bold and correspond to the Mixture VRNN-PI. Additionally, all VRNN-PI variants outperform the RNN baseline in each feature distance with the exception of discrete VRNN-PI on right wing length.

mostly idle behavior, which further concentrates the feature distributions onto the mode. Despite that, we find that all variants of VRNN-PI agree with the real data better than the RNN, with Mixture VRNN-PI being clearly the best. Table **??** provides a quantitative summary of these results.

Our third and final experiment investigates the quality of uncertainty estimates produced by various models. For this purpose we again seed the model with an initial sequence of actions, then see how the probability mass over the flies' future positions spreads over time and compare it with the actual positions in the dataset. Figure 5 visualizes the results. Again we find that Mixture VRNN-PI performs best, followed by the other VRNN-PI variants and then by the RNN.

5 Discussion

We have shown that deep generative models can be successfully used to perform imitation learning in biological systems in an ego-centric setting. These results are useful both as a practical imitation learning tool and as an approach for building theoretical models of how animals make decisions. We have demonstrated better performance than the state-of-the-art RNN, both in terms of generating more realistic behavior and in terms of providing better calibrated uncertainty estimates. The latter is particularly useful in tasks that use such learned model for decision making, for example by constructing confidence regions of where the tracked agent will be located at a certain time in future. We can envision diverse uses for this information, for example in systems tracking wild animals that warn people when their settlements are being approached and in autonomous vehicles that need to be sure that pedestrians will not jump in front of it.

A particularly interesting result we obtained is that including discrete latent variables appears to significantly improve performance when adding more continuous latent variables does not. If this result is confirmed in analysis of different datasets, it would have important implications for the construction of optimization algorithms for deep generative models. In particular it would make a strong case for devising variance reduction methods that work when the reparameterization trick is not applicable.

As we are directly modeling biological agents, we might also speculate about what our results indicate about how real animal brains work. In particular the better performance of VRNN compared with RNN might suggest that stochastic aspects of animal behavior are better explained by the state of the brain itself evolving stochastically, rather than purely by stochasticity in the translation of brain state to actions. That the inclusion of discrete latent variables increases performance may lead us to speculate that the brain actually encodes some of the information inside it stochastically. Of course, making such claims would require enormous amounts of evidence that we are not even beginning to supply. All the same, we find the idea captivating.

We believe that our observation of obtaining better performance using a mixture of discrete and continuous latent variables warrants further investigations. Alternative avenues for future work may include using joint modeling of x and y (Vedantam et al., 2017), adding more powerful inference networks, such as normalizing flows (Rezende and Mohamed, 2015), and also improving on the model towards representation learning of interpretable latents (Alemi et al., 2018). Finally, the temporal memory mechanism used an RNN, which is known to capture long-range dependencies up to 200-300 timesteps. This may also be replaced with more powerful memory-augmented recurrent neural network models in our framework, e.g. differentiable neural computers (DNC) or neural Turing machines (NTM) (Gemici et al., 2017).

References

Abbeel, P. and Ng, A. Y. 2004. Apprenticeship learning via inverse reinforcement learning. *ICML.*

Alemi, A., Poole, B., Fischer, I., Dillon, J., Saurous, R. A., and Murphy, K. 2018. Fixing a broken elbo. In *International Conference on Machine Learning*, pages 159–168.

Bayley, H. and Cremer, P. S. 2001. Stochastic sensors inspired by biology. *Nature*, 413(6852):226.

Bornschein, J. and Bengio, Y. 2014. Reweighted wake-sleep. *arXiv preprint arXiv:1406.2751*.

Burda, Y., Grosse, R., and Salakhutdinov, R. 2015. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*.

Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A. C., and Bengio, Y. 2015. A recurrent latent variable model for sequential data. In *Advances in neural information processing systems*, pages 2980–2988.

Clandinin, T. R. and Giocomo, L. M. 2015. Neuroscience: Internal compass puts flies in their place. *Nature*, 521(7551):165.

Cueva, C. J. and Wei, X.-X. 2018. Emergence of grid-like representations by training recurrent neural networks to perform spatial localization. *arXiv preprint arXiv:1803.07770*.

Eyjolfsdottir, E., Branson, K., Yue, Y., and Perona, P. 2016. Learning recurrent representations for hierarchical behavior modeling. In *International Conference on Learning Representations*.

Eyjolfsdottir, E., Branson, S., Burgos-Artizzu, X. P., Hoopfer, E. D., Schor, J., Anderson, D. J., and Perona, P. 2014. Detecting social actions of fruit flies. In *European Conference on Computer Vision*, pages 772–787. Springer.

Gemici, M., Hung, C.-C., Santoro, A., Wayne, G., Mohamed, S., Rezende, D. J., Amos, D., and Lillicrap, T. 2017. Generative temporal models with memory. *arXiv preprint arXiv:1702.04649*.

Grathwohl, W., Choi, D., Wu, Y., Roeder, G., and Duvenaud, D. 2017. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. *arXiv preprint arXiv:1711.00123*.

Jang, E., Gu, S., and Poole, B. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Johnson, M., Duvenaud, D. K., Wiltschko, A., Adams, R. P., and Datta, S. R. 2016. Composing graphical models with neural networks for structured representations and fast inference. In *Advances in neural information processing systems*, pages 2946–2954.

Kaelbling, L. P. 1993. Hierarchical learning in stochastic domains: Preliminary results. In *Proceedings of the tenth international conference on machine learning*, volume 951, pages 167–173.

Kingma, D. P. and Welling, M. 2013. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*.

Le, T. A., Kosiorek, A. R., Siddharth, N., Teh, Y. W., and Wood, F. 2018. Revisiting reweighted wake-sleep. *arXiv preprint arXiv:1805.10469*.

Maddison, C. J., Mnih, A., and Teh, Y. W. 2016. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*.

Mnih, A. and Gregor, K. 2014. Neural variational inference and learning in belief networks. *arXiv preprint arXiv:1402.0030*.

Mnih, A. and Rezende, D. J. 2016. Variational inference for monte carlo objectives. *arXiv preprint arXiv:1602.06725*.

Osa, T., Pajarinen, J., Neumann, G., Bagnell, J. A., Abbeel, P., and Peters, J. 2018. *An Algorithmic Perspective on Imitation Learning*. Foundations and Trends in Robotics.

Rainforth, T., Kosiorek, A. R., Le, T. A., Maddison, C. J., Igl, M., Wood, F., and Teh, Y. W. 2018. Tighter variational bounds are not necessarily better. In *International Conference on Machine Learning*.

Rezende, D. J. and Mohamed, S. 2015. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*.

Schaal, S. 1999. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, 3(6):233–242.

Sermanet, P., Xu, K., and Levine, S. 2016. Unsupervised perceptual rewards for imitation learning. *arXiv preprint arXiv:1612.06699*.

Sutton, R. S. and Barto, A. G. 2018. *Reinforcement Learning: An Introduction*. MIT Press.

Tucker, G., Mnih, A., Maddison, C. J., Lawson, J., and Sohl-Dickstein, J. 2017. Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models. In *Advances in Neural Information Processing Systems*, pages 2627–2636.

Vedantam, R., Fischer, I., Huang, J., and Murphy, K. 2017. Generative models of visually grounded imagination. *arXiv preprint arXiv:1705.10762*.

Wen, T. H., Gasic, M., Mrksic, N., Su, P.-H., Vandyke, D., and Young, S. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *EMNLP*.

Zhan, E., Zheng, S., Yue, Y., Sha, L., and Lucey, P. 2019. Generating multi-agent trajectories using programmatic weak supervision. *ICLR*.