Learning Calibratable Policies using Programmatic Style-Consistency

Eric Zhan [1] **Albert Tseng** [1] **Yisong Yue** [1] **Adith Swaminathan** [2] **Matthew Hausknecht** [2]

Abstract

We study the problem of controllable generation of long-term sequential behaviors, where the goal is to calibrate to multiple behavior styles simultaneously. In contrast to the well-studied areas of controllable generation of images, text, and speech, there are two questions that pose significant challenges when generating long-term behaviors: how should we specify the factors of variation to control, and how can we ensure that the generated behavior faithfully demonstrates combinatorially many styles? We leverage programmatic labeling functions to specify controllable styles, and derive a formal notion of style-consistency as a learning objective, which can then be solved using conventional policy learning approaches. We evaluate our framework using demonstrations from professional basketball players and agents in the MuJoCo physics environment, and show that existing approaches that do not explicitly enforce style-consistency fail to generate diverse behaviors whereas our learned policies can be calibrated for up to $4^5(1024)$ distinct style combinations.

1. Introduction

The widespread availability of recorded tracking data is enabling the study of complex behaviors in many domains, including sports (Chen et al., 2016a; Le et al., 2017b; Zhan et al., 2019; Yeh et al., 2019), video games (Kurin et al., 2017; Broll et al., 2019; Hofmann, 2019), laboratory animals (Eyjolfsdottir et al., 2014; 2017; Branson et al., 2009; Johnson et al., 2016), facial expressions (Suwajanakorn et al., 2017; Taylor et al., 2017), commonplace activities such as cooking (Nishimura et al., 2019), and transportation (Bojarski et al., 2016; Luo et al., 2018; Li et al., 2018; Chang et al., 2019). A key aspect of modern behavioral datasets is that the behaviors can exhibit very diverse styles (e.g., from multiple demonstrators). For example, Figure 1a depicts demonstrations from basketball players with variations in speed, desired destinations, and curvature of movement.

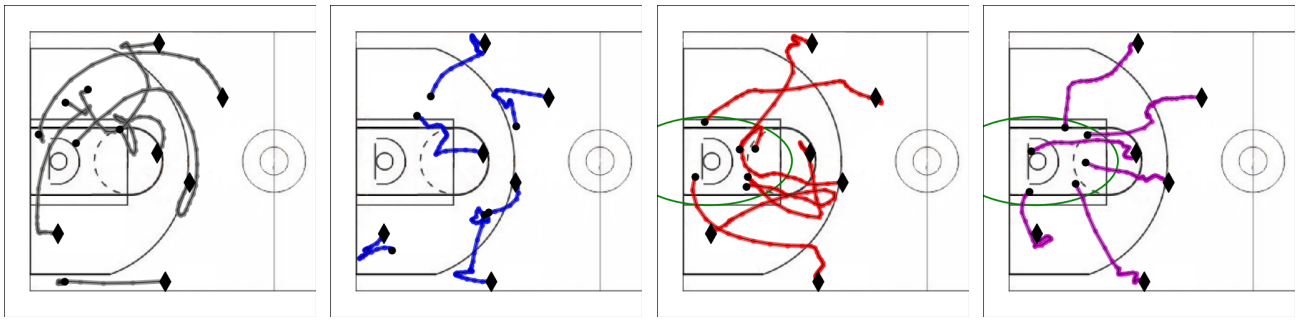
The goal of this paper is to study controllable generation of diverse behaviors by learning to imitate raw demonstrations; or more technically, to develop style-calibrated imitation learning methods. A controllable, or calibratable, policy would enable the generation of behaviors consistent with various styles, such as low movement speed (Figure 1b), or approaching the basket (Figure 1c), or both styles simultaneously (Figure 1d). Style-calibrated imitation learning methods that can yield such policies can be broadly useful to: (a) perform more robust imitation learning from diverse demonstrations (Wang et al., 2017; Broll et al., 2019), (b) enable diverse exploration in reinforcement learning agents (Co-Reyes et al., 2018), or (c) visualize and extrapolate counterfactual behaviors beyond those seen in the dataset (Le et al., 2017a), amongst many other tasks.

Performing style-calibrated imitation is a challenging task. First, what constitutes a "style"? Second, when can we be certain that a policy is "calibrated" when imitating a style? Third, how can we scale policy learning to faithfully generate combinatorially many styles? In related tasks like controllable image generation, common approaches for calibration use adversarial information factorization or mutual information between generated images and user-specified styles (e.g. gender, hair length, etc.) (Creswell et al., 2017; Lample et al., 2017; Chen et al., 2016b). However, we find that these *indirect* approaches fall well short of generating calibratable sequential behaviors. Intuitively, the aforementioned objectives provide only indirect proxies for style-calibration. For example, Figure 2 illustrates that an indirect baseline approach struggles to reliably generate trajectories to reach a certain displacement, even though the dataset contains many examples of such behavior.

Research questions. We seek to answer three research questions while tackling this challenge. The first is strategic: since high-level stylistic attributes like movement speed are typically not provided with the raw demonstration data, what systematic form of domain knowledge can we leverage to quickly and cleanly extract highly varied style information from raw behavioral data? The second is formulaic: how can

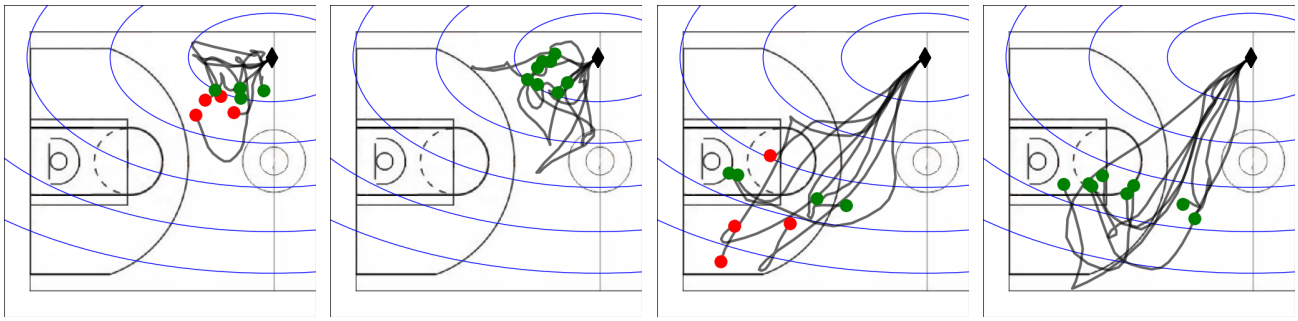
[1]California Institute of Technology, Pasadena, CA [2]Microsoft Research, Redmond, WA. Correspondence to: Eric Zhan <ezhan@caltech.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020.



(a) Expert demonstrations (b) Style: SPEED (c) Style: DESTINATION (d) Both styles

Figure 1. Basketball trajectories from policies that are: (a) the expert; (b) calibrated to move at low speeds; (c) calibrated to end near the basket (within green boundary); and (d) calibrated for both (b,c) simultaneously. Diamonds (⧫) and dots (●) are initial and final positions.



(a) Baseline, low displacement (b) Ours, low displacement (c) Baseline, high displacement (d) Ours, high displacement

Figure 2. Basketball trajectories sampled from baseline policies and our models calibrated to the style of `DISPLACEMENT` with 6 classes corresponding to regions separated by blue lines. Diamonds (⧫) and dots (●) indicate initial and final positions respectively. Each policy is conditioned on a label class for `DISPLACEMENT` (low in (a,b), high in (c,d)). Green dots indicate trajectories that are consistent with the style label, while red dots indicate those that are not. Our policy (b,d) is better calibrated for this style than the baselines (a,c).

we formalize the learning objective to encourage learning style-calibratable policies that can be controlled to realize many diverse styles? The third is algorithmic: how do we design practical learning approaches that reliably optimize the learning objective?

Our contributions. To address these questions, we present a novel framework inspired by *data programming* (Ratner et al., 2016), a paradigm in weak supervision that utilizes automated labeling procedures, called labeling functions, to learn without ground-truth labels. In our setting, labeling functions enable domain experts to quickly translate domain knowledge of diverse styles into programmatically generated style annotations. For instance, it is trivial to write programmatic labeling functions for the styles depicted in Figures 1 & 2 (speed and destination). Labeling functions also motivate a new learning objective, which we call *programmatic style-consistency*: rollouts generated by a policy calibrated for a particular style should return the same style label when fed to the programmatic labeling function. This notion of style-consistency provides a *direct* approach to measuring how calibrated a policy is, and does not suffer from the weaknesses of indirect approaches such as mutual information estimation. In the basketball example of scoring when near the basket, trajectories that perform correlated events (like turning towards the basket) will not return the desired style label when fed to the labeling function that checks for scoring events. We elaborate on this in Section 4.

We demonstrate style-calibrated policy learning in Basketball and MuJoCo domains. Our experiments highlight the modularity of our approach – we can plug-in any policy class and any imitation learning algorithm and reliably optimize for style-consistency using the approach of Section 5. The resulting learned policies can achieve very fine-grained and diverse style-calibration with negligible degradation in imitation quality – for example, our learned policy is calibrated to $4^5(1024)$ distinct style combinations in Basketball.

2. Related Work

Our work combines ideas from policy learning and data programming to develop a weakly supervised approach for more explicit and fine-grained calibration. As such, our work is related to learning disentangled representations and controllable generative modeling, reviewed below.

Imitation learning of diverse behaviors has focused on unsupervised approaches to infer latent variables/codes that

capture behavior styles (Li et al., 2017; Hausman et al., 2017; Wang et al., 2017). Similar approaches have also been studied for generating text conditioned on attributes such as sentiment or tense (Hu et al., 2017). A typical strategy is to maximize the mutual information between the latent codes and trajectories, in contrast to our notion of programmatic style-consistency.

Disentangled representation learning aims to learn representations where each latent dimension corresponds to exactly one desired factor of variation (Bengio et al., 2012). Recent studies (Locatello et al., 2019) have noted that popular techniques (Chen et al., 2016b; Higgins et al., 2017; Kim & Mnih, 2018; Chen et al., 2018) can be sensitive to hyperparameters and that evaluation metrics can be correlated with certain model classes and datasets, which suggests that fully unsupervised learning approaches may, in general, be unreliable for discovering cleanly calibratable representations. We avoid this roadblock by relying on programmatic labeling functions to provide weak supervision.

Conditional generation for images has recently focused on *attribute manipulation* (Bao et al., 2017; Creswell et al., 2017; Klys et al., 2018), which aims to enforce that changing a label affects only one aspect of the image (similar to disentangled representation learning). We extend these models and compare with our approach in Section 6. Our experiments suggest that these algorithms do not necessarily scale well into sequential domains.

Enforcing consistency in generative modeling, such as cycle-consistency in image generation (Zhu et al., 2017), and self-consistency in hierarchical reinforcement learning (Co-Reyes et al., 2018) has proved beneficial. The former minimizes a discriminative disagreement, whereas the latter minimizes a distributional disagreement between two sets of generated behaviors (e.g., KL-divergence). From this perspective, our style-consistency notion is more similar to the former; however we also enforce consistency over multiple time-steps, which is more similar to the latter.

Goal-conditioned policy learning considers policies that take as input the current state along with a desired goal state (e.g., a location), and then must execute a sequence of actions to achieve the goal states. In some cases, the goal states are provided exogenously (Zheng et al., 2016; Le et al., 2018; Broll et al., 2019; Ding et al., 2019), and in other cases the goal states are learned as part of a hierarchical policy learning approach (Co-Reyes et al., 2018; Sharma et al., 2020) in a way that uses a self-consistency metric similar to our style-consistency approach. Our approach can be viewed as complementary to these approaches as the goal is to study more general notions of consistency (e.g., our styles subsume goals as a special case) as well as to scale to combinatorial joint style spaces.

Hierarchical control via learning latent motor dynamics is concerned with recovering a latent representation of motor control dynamics such that one can easily design controllers in the latent space (which then get decoded into actions). The high level controllers can then be designed afterwards in a pipelined workflow (Losey et al., 2020; Ling et al., 2020; Luo et al., 2020). The controllers are effective for short time horizons and focus on finding good representations of complex dynamics, whereas we focus on controlling behavior styles that can span longer horizons.

3. Background: Imitation Learning for Behavior Trajectories

Since our focus is on learning style-calibratable generative policies, for simplicity we develop our approach with the basic imitation learning paradigm of behavioral cloning. Interesting future directions include composing our approach with more advanced imitation learning approaches like DAGGER (Ross et al., 2011), GAIL (Ho & Ermon, 2016) as well as with reinforcement learning.

Notation. Let $\mathcal{S}$ and $\mathcal{A}$ denote the environment state and action spaces. At each timestep t, an agent observes state $\mathbf{s}_t \in \mathcal{S}$ and executes action $\mathbf{a}_t \in \mathcal{A}$ using a policy $\pi : \mathcal{S} \to \mathcal{A}$. The environment then transitions to the next state $\mathbf{s}_{t+1}$ according to a (typically unknown) dynamics function $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$. For the rest of this paper, we assume f is deterministic; a modification of our approach for stochastic f is included in Appendix B. A trajectory τ is a sequence of T state-action pairs and the last state: $\tau = \{(\mathbf{s}_t, \mathbf{a}_t)\}_{t=1}^{T} \cup \{\mathbf{s}_{T+1}\}$. Let $\mathcal{D}$ be a set of N trajectories collected from expert demonstrations. In our experiments, each trajectory in $\mathcal{D}$ has the same length T, but in general this does not need to be the case.

Learning objective. We begin with the basic imitation learning paradigm of behavioral cloning (Syed & Schapire, 2008). The goal is to learn a policy that behaves like the pre-collected demonstrations:

$$\pi^* = \arg\min_{\pi} \mathbb{E}_{\tau \sim \mathcal{D}} \left[\mathcal{L}^{\text{imitation}}(\tau, \pi) \right], \tag{1}$$

where $\mathcal{L}^{\text{imitation}}$ is a loss function that quantifies the mismatch between actions chosen by π and those in the demonstrations. Since we are primarily interested in probabilistic or generative policies, we typically use (variants of) negative log-density: $\mathcal{L}(\tau, \pi) = \sum_{t=1}^{T} -\log \pi(\mathbf{a}_t | \mathbf{s}_t)$, where $\pi(\mathbf{a}_t | \mathbf{s}_t)$ is the probability of π picking action $\mathbf{a}_t$ in $\mathbf{s}_t$.

Policy class of π. Common model choices for instantiating π include sequential generative models like recurrent Neural Networks (RNN) and trajectory variational autoencoders (TVAE). TVAEs introduce a latent variable $\mathbf{z}$ (also called a trajectory embedding), an encoder network q_ϕ, a policy

decoder π_θ, and a prior distribution p on $\mathbf{z}$. They have been shown to work well in a range of generative policy learning settings (Wang et al., 2017; Ha & Eck, 2018; Co-Reyes et al., 2018), and have the following imitation learning objective:

$$\mathcal{L}^{\text{tvae}}(\tau, \pi_\theta; q_\phi) = \mathbb{E}_{q_\phi(\mathbf{z}|\tau)}\left[\sum_{t=1}^{T} -\log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t, \mathbf{z})\right] + D_{KL}\big(q_\theta(\mathbf{z}|\tau)||p(\mathbf{z})\big). \quad (2)$$

The first term in (2) is the standard *negative log-density* that the policy assigns to trajectories in the dataset, while the second term is the *KL-divergence* between the prior and approximate posterior of trajectory embeddings $\mathbf{z}$. The main shortcoming of TVAEs and related approaches, which we address in Sections 4 & 5, is that the resulting policies cannot be easily calibrated to generate specific styles. For instance, the goal of the trajectory embedding $\mathbf{z}$ is to capture all the styles that exist in the expert demonstrations, but there is no guarantee that the embeddings cleanly encode the desired styles in a calibrated way. Previous work has largely relied on unsupervised learning techniques that either require significant domain knowledge (Le et al., 2017b), or have trouble scaling to complex styles commonly found in real-world applications (Wang et al., 2017; Li et al., 2017).

4. Programmatic Style-consistency

Building upon the basic setup in Section 3, we focus on the setting where the demonstrations $\mathcal{D}$ contain diverse behavior styles. To start, let $\mathbf{y} \in Y$ denote a single style label (e.g., speed or destination, as shown in Figure 1). Our goal is to learn a policy π that can be explicitly calibrated to $\mathbf{y}$, i.e., trajectories generated by $\pi(\cdot|\mathbf{y})$ should match the demonstrations in $\mathcal{D}$ that exhibit style $\mathbf{y}$.

Obtaining style labels can be expensive using conventional annotation methods, and unreliable using unsupervised approaches. We instead utilize easily programmable labeling functions that automatically produce style labels. We then formalize a notion of style-consistency as a learning objective, and in Section 5 describe a practical learning approach.

Labeling functions. Introduced in the data programming paradigm (Ratner et al., 2016), labeling functions programmatically produce weak and noisy labels to learn models on otherwise unlabeled datasets. A significant benefit is that labeling functions are often simple scripts that can be quickly applied to the dataset, which is much cheaper than manual annotations and more reliable than unsupervised methods. In our framework, we study behavior styles that can be represented as labeling functions, which we denote λ, that map trajectories τ to style labels $\mathbf{y}$. For example:

$$\lambda(\tau) = \mathbb{1}\{\|\mathbf{s}_{T+1} - \mathbf{s}_1\|_2 > c\}, \quad (3)$$

which distinguishes between trajectories with large (greater than a threshold c) versus small total displacement. We experiment with a range of labeling functions, as described in Section 6. Many behavior styles used in previous work can be represented as labeling functions, e.g., agent speed (Wang et al., 2017). Multiple labeling functions can be provided at once resulting in a combinatorial space of joint style labels. We use trajectory-level labels $\lambda(\tau)$ in our experiments, but in general labeling functions can be applied on subsequences $\lambda(\tau_{t:t+h})$ to obtain per-timestep labels, e.g., agent goal (Broll et al., 2019). We can efficiently annotate datasets using labeling functions, which we denote as $\lambda(\mathcal{D}) = \{(\tau_i, \lambda(\tau_i))\}_{i=1}^N$. Our goal can now be phrased as: given $\lambda(\mathcal{D})$, train a policy $\pi : \mathcal{S} \times Y \mapsto \mathcal{A}$ such that $\pi(\cdot|\mathbf{y})$ is calibrated to styles $\mathbf{y}$ found in $\lambda(\mathcal{D})$.

Style-consistency. A key insight in our work is that labeling functions naturally induce a metric for calibration. If a policy $\pi(\cdot|\mathbf{y})$ is calibrated to λ, we would expect the generated behaviors to be consistent with the label. So, we expect the following loss to be small:

$$\mathbb{E}_{\mathbf{y}\sim p(\mathbf{y}), \tau\sim\pi(\cdot|\mathbf{y})}\Big[\mathcal{L}^{\text{style}}\big(\lambda(\tau), \mathbf{y}\big)\Big], \quad (4)$$

where $p(\mathbf{y})$ is a prior over the style labels, and τ is obtained by executing the style-conditioned policy in the environment. $\mathcal{L}^{\text{style}}$ is thus a disagreement loss over labels that is minimized at $\lambda(\tau) = \mathbf{y}$, e.g., $\mathcal{L}^{\text{style}}\big(\lambda(\tau), \mathbf{y}\big) = \mathbb{1}\{\lambda(\tau) \neq \mathbf{y}\}$ for categorical labels. We refer to (4) as the *style-consistency* loss, and say that $\pi(\cdot|\mathbf{y})$ is maximally calibrated to λ when (4) is minimized. Our learning objective adds (1) with (4):

$$\pi^* = \arg\min_\pi \mathbb{E}_{\big(\tau,\lambda(\tau)\big)\sim\lambda(\mathcal{D})}\Big[\mathcal{L}^{\text{imitation}}\Big(\tau, \pi\big(\cdot \mid \lambda(\tau)\big)\Big)\Big] + \mathbb{E}_{\mathbf{y}\sim p(\mathbf{y}), \tau\sim\pi(\cdot|\mathbf{y})}\Big[\mathcal{L}^{\text{style}}\big(\lambda(\tau), \mathbf{y}\big)\Big]. \quad (5)$$

The simplest choice for the prior distribution $p(\mathbf{y})$ is the marginal distribution of styles in $\lambda(\mathcal{D})$. The first term in (5) is a standard imitation learning objective and can be tractably estimated using $\lambda(\mathcal{D})$. To enforce style-consistency with the second term, conceptually we need to sample several $\mathbf{y} \sim p(\mathbf{y})$, then several rollouts $\tau \sim \pi(\cdot \mid \mathbf{y})$ from the current policy, and query the labeling function for each of them. Furthermore, if λ is a non-differentiable function defined over the entire trajectory, as is the case in (3), then we cannot simply backpropagate the style-consistency loss. In Section 5, we introduce differentiable approximations to more easily optimize the objective in (5).

Combinatorial joint style space. Our notion of style-consistency can be easily extended to optimize for combinatorially-many joint styles when multiple labeling functions are provided. Suppose we have M labeling functions $\{\lambda_i\}_{i=1}^M$ and corresponding label spaces $\{Y_i\}_{i=1}^M$. Let λ denote $(\lambda_1, \ldots, \lambda_M)$ and $\mathbf{y}$ denote $(\mathbf{y}_1, \ldots, \mathbf{y}_M)$. Style-

consistency loss becomes:

$$\mathbb{E}_{\mathbf{y}\sim p(\mathbf{y}),\tau\sim\pi(\cdot|\mathbf{y})}\left[\sum_{i=1}^{M}\mathcal{L}_i^{\text{style}}\big(\lambda_i(\tau),\mathbf{y}_i\big)\right]. \tag{6}$$

Note that style-consistency is optimal when the generated trajectory agrees with *all* labeling functions. Although challenging to achieve, this outcome is most desirable, i.e. $\pi(\cdot|\mathbf{y})$ is calibrated to *all* styles simultaneously. Indeed, a key metric that we evaluate is how well various learned policies can be calibrated to all styles simultaneously (i.e., loss of 0 only if all styles are calibrated, and loss of 1 otherwise).

5. Learning Approach

Optimizing (5) is challenging due to the long-time horizon and non-differentiability of the labeling functions λ.[1] Given unlimited queries to the environment, one could naively employ model-free reinforcement learning, e.g., estimating (4) using rollouts and optimizing using policy gradient approaches. We instead take a model-based approach, described generically in Algorithm 1, that is more computationally-efficient and decomposable (i.e., transparent). The model-based approach is compatible with batch or offline learning, and we found it particularly useful for diagnosing deficiencies in our algorithmic framework. We first introduce a label approximator for λ, and then show how to optimize through the environmental dynamics using a differentiable model-based learning approach.

Approximating labeling functions. To deal with non-differentiability of λ, we approximate it with a differentiable function C_ψ^λ parameterized by ψ:

$$\psi^* = \arg\min_{\psi}\mathbb{E}_{\left(\tau,\lambda(\tau)\right)\sim\lambda(\mathcal{D})}\left[\mathcal{L}^{\text{label}}\big(C_\psi^\lambda(\tau),\lambda(\tau)\big)\right] \tag{7}$$

Here, $\mathcal{L}^{\text{label}}$ is a differentiable loss that approximates $\mathcal{L}^{\text{style}}$, such as cross-entropy loss when $\mathcal{L}^{\text{style}}$ is the $0/1$ loss. In our experiments we use a RNN to represent C_ψ^λ. We then modify the style-consistency term in (5) with $C_{\psi^*}^\lambda$ and optimize:

$$\begin{aligned}\pi^* = \arg\min_{\pi}\mathbb{E}_{\left(\tau,\lambda(\tau)\right)\sim\lambda(\mathcal{D})}&\left[\mathcal{L}^{\text{imitation}}\Big(\tau,\pi\big(\cdot\mid\lambda(\tau)\big)\Big)\right]\\ &+\mathbb{E}_{\mathbf{y}\sim p(\mathbf{y}),\tau\sim\pi(\cdot|\mathbf{y})}\left[\mathcal{L}^{\text{label}}\big(C_{\psi^*}^\lambda(\tau),\mathbf{y}\big)\right].\end{aligned} \tag{8}$$

Optimizing $\mathcal{L}^{\text{style}}$ over trajectories. The next challenge is to optimize style-consistency over multiple time steps. Consider the labeling function in (3) that computes the difference between the first and last states. Our label approximator $C_{\psi^*}^\lambda$ may converge to a solution that ignores all inputs except for $\mathbf{s}_1$ and $\mathbf{s}_{T+1}$. In this case, $C_{\psi^*}^\lambda$ provides no learning signal about intermediate steps. As such, effective optimization of style-consistency in (8) requires informative learning signals on all actions at every step, which can be viewed as a type of credit assignment problem.

[1] This issue is not encountered in previous work on style-dependent imitation learning (Li et al., 2017; Hausman et al., 2017), since they use purely unsupervised methods such as maximizing mutual information which is differentiable.

Algorithm 1 Generic recipe for optimizing (5)

1: **Input**: demonstrations $\mathcal{D}$, labeling functions λ
2: construct $\lambda(\mathcal{D})$ by applying λ on trajectories in $\mathcal{D}$
3: optimize (7) to convergence to learn $C_{\psi^*}^\lambda$
4: optimize (8) to convergence to learn π^*

In general, model-free and model-based approaches address this challenge in dramatically different ways and for different problem settings. A model-free solution views this credit assignment challenge as analogous to that faced by reinforcement learning (RL), and repurposes generic reinforcement learning algorithms. Crucially, they assume access to the environment to collect more rollouts under any new policy. A model-based solution does not assume such access and can operate only with the batch of behavior data $\mathcal{D}$; however they can have an additional failure mode since the learned models may provide an inaccurate signal for proper credit assignment. We choose a model-based approach, while exploiting access to the environment when available to refine the learned models, for two reasons: (a) we found it to be compositionally simpler and easier to debug; and (b) we can use the learned model to obtain hallucinated rollouts of any policy efficiently during training.

Modeling dynamics for credit assignment. Our model-based approach utilizes a dynamics model M_φ to approximate the environment's dynamics by predicting the change in state given the current state and action:

$$\varphi^* = \arg\min_{\varphi}\mathbb{E}_{\tau\sim\mathcal{D}}\sum_{t=1}^{T}\mathcal{L}^{\text{dynamics}}\big(M_\varphi(\mathbf{s}_t,\mathbf{a}_t),(\mathbf{s}_{t+1}-\mathbf{s}_t)\big), \tag{9}$$

where $\mathcal{L}^{\text{dynamics}}$ is often L_2 or squared-L_2 loss (Nagabandi et al., 2018; Luo et al., 2019). This allows us to generate trajectories by rolling out: $\mathbf{s}_{t+1}=\mathbf{s}_t+M_\varphi\big(\mathbf{s}_t,\pi(\mathbf{s}_t)\big)$. Then optimizing for style-consistency in (8) would backpropagate through our dynamics model M_φ and provide informative learning signals to the policy at every timestep.

We outline our model-based approach in Algorithm 2. Lines 12-15 describe an optional step to fine-tune the dynamics model by querying the environment using the current policy (similar to Luo et al. (2019)); we found that this can improve style-consistency in some experiments. In Appendix B we elaborate how the dynamics model and objective of Eqn (9) is changed if the environment is stochastic.

Discussion. To summarize, we claim that style-consistency

Algorithm 2 Model-based approach for Algorithm 1

Input: demonstrations $\mathcal{D}$, labeling function λ, label approximator C^{λ}_{ψ}, dynamics model M_{φ}
$\lambda(\mathcal{D}) \leftarrow \{(\tau_i, \lambda(\tau_i))\}_{i=1}^{N}$
for n_{dynamics} iterations **do**
  optimize (9) with batch from $\mathcal{D}$
end for
for n_{label} iterations **do**
  optimize (7) with batch from $\lambda(\mathcal{D})$
end for
for n_{policy} iterations **do**
  $\mathcal{B} \leftarrow \{$ n_{collect} trajectories using M_{φ} and π $\}$
  optimize (8) with batch from $\lambda(\mathcal{D})$ and $\mathcal{B}$
  for n_{env} iterations **do**
    $\tau_{\text{env}} \leftarrow \{$ 1 trajectory using environment and π $\}$
    optimize (9) with τ_{env}
  end for
end for

is an "objective" metric to measure the quality of calibration. Our learning approach uses off-the-shelf methods to enforce style-consistency during training. We anticipate several variants of style-consistent policy learning of Algorithm 1 – for example, using model-free RL, using environment/model rollouts to fine-tune the labeling function approximator, using style-conditioned policy classes, or using other loss functions to encourage imitation quality. Our experiments in Section 6 establish that our style-consistency loss provides a clear learning signal, that no prior approach directly enforces this consistency, and that our approach accomplishes calibration for a combinatorial joint style space.

6. Experiments

We first briefly describe our experimental setup and baseline choices, and then discuss our main experimental results. A full description of experiments is available in Appendix C.[2]

Data. We validate our framework on two datasets: 1) a collection of professional basketball player trajectories with the goal of learning a policy that generates realistic player-movement, and 2) a Cheetah agent running horizontally in MuJoCo (Todorov et al., 2012) with the goal of learning a policy with calibrated gaits. The former has a known dynamics function: $f(\mathbf{s}_t, \mathbf{a}_t) = \mathbf{s}_t + \mathbf{a}_t$, where $\mathbf{s}_t$ and $\mathbf{a}_t$ are the player's position and velocity on the court respectively; we expect the dynamics model M_{φ} to easily recover this function. The latter has an unknown dynamics function (which we learn a model of when approximating style-consistency). We obtain Cheetah demonstrations from a collection of policies trained using `pytorch-a2c-ppo-acktr` (Kostrikov, 2018) to interface with the DeepMind Control Suite's Cheetah domain (Tassa et al., 2018)—see Appendix C for details.

Labeling functions. Labeling functions for Basketball include: 1) average `SPEED` of the player, 2) `DISPLACEMENT` from initial to final position, 3) distance from final position to a fixed `DESTINATION` on the court (e.g. the basket), 4) mean `DIRECTION` of travel, and 5) `CURVATURE` of the trajectory, which measures the player's propensity to change directions. For Cheetah, we have labeling functions for the agent's 1) `SPEED`, 2) `TORSO HEIGHT`, 3) `BACK-FOOT HEIGHT`, and 4) `FRONT-FOOT HEIGHT` that can be trivially computed from trajectories extracted from the environment.

We threshold the aforementioned labeling functions into categorical labels (leaving real-valued labels for future work) and use (4) for style-consistency with $\mathcal{L}^{\text{style}}$ as the $0/1$ loss. We use cross-entropy for $\mathcal{L}^{\text{label}}$ and list all other hyperparameters in Appendix C.

Metrics. We will primarily study two properties of the learned models in our experiments – imitation quality, and style-calibration quality. For measuring imitation quality of generative models, we report the *negative log-density* term in (2), also known as the reconstruction loss term in VAE literature (Kingma & Welling, 2014; Ha & Eck, 2018), which corresponds to how well the policy can reconstruct trajectories from the dataset.

To measure style-calibration, we report style-consistency results as $1 - \mathcal{L}^{\text{style}}$ in (4) so that all results are easily interpreted as accuracies. In Section 6.5, we find that style-consistency indeed captures a reasonable notion of calibration – when the labeling function is inherently noisy and style-calibration is hard, style-consistency correspondingly decreases. In Section 6.3, we find that the goals of imitation (as measured by negative log-density) and calibration (as measured by style-consistency) may not always be aligned – investigating this trade-off is an avenue for future work.

Baselines. Our main experiments use TVAEs as the underlying policy class. In Section 6.4, we also experiment with an RNN policy class. We compare our approach, CTVAE-style, with 3 baselines:

1. **CTVAE**: conditional TVAEs (Wang et al., 2017).
2. **CTVAE-info**: CTVAE with information factorization (Creswell et al., 2017), *indirectly* maximizes style-consistency by removing correlation of $\mathbf{y}$ with $\mathbf{z}$.
3. **CTVAE-mi**: CTVAE with mutual information maximization between style labels and trajectories. This is a supervised variant of unsupervised models (Chen et al., 2016b; Li et al., 2017), and also requires learning a dynamics model for sampling policy rollouts.

[2] Code is available at: https://github.com/ezhan94/calibratable-style-consistency.

Detailed descriptions of baselines are in Appendix A. All baseline models build upon TVAEs, which are also conditioned on a latent variable (see Section 3) and only fundamentally differ in how they encourage the calibration of policies to different style labels. We highlight that the underlying model choice is orthogonal to our contributions; our framework is compatible with other policy models (see Section 6.4).

Model details. We model all trajectory embeddings $\mathbf{z}$ as a diagonal Gaussian with a standard normal prior. Encoder q_ϕ and label approximators C_ψ^λ are bi-directional GRUs (Cho et al., 2014) followed by linear layers. Policy π_θ is recurrent for basketball, but non-recurrent for Cheetah. The Gaussian log sigma returned by π_θ is state-dependent for basketball, but state-independent for Cheetah. For Cheetah, we made these choices based on prior work in MuJoCo for training gait policies (Wang et al., 2017). For Basketball, we observed a lot more variation in the 500k demonstrations so we experimented with a more flexible model. See Appendix C for hyperparameters.

6.1. How well can we calibrate policies for single styles?

We first threshold labeling functions into 3 classes for Basketball and 2 classes for Cheetah; the marginal distribution $p(\mathbf{y})$ of styles in $\lambda(\mathcal{D})$ is roughly uniform over these classes. Then we learn a policy π^* calibrated to each of these styles. Finally, we generate rollouts from each of the learned policies to measure style-consistency. Table 1 compares the median style-consistency (over 5 seeds) of learned policies. For Basketball, CTVAE-style significantly outperforms baselines and achieves almost perfect style-consistency for 4 of the 5 styles. For Cheetah, CTVAE-style outperforms all baselines, but the absolute performance is lower than for Basketball – we conjecture that this is due to the complex environment dynamics that can be challenging for model-based approaches. Figure 5 in Appendix D shows a visualization of our CTVAE-style policy calibrated for `DESTINATION(net)`.

We also consider cases in which labeling functions can have several classes and non-uniform distributions (i.e. some styles are more/less common than others). We threshold `DISPLACEMENT` into 6 classes for Basketball and `SPEED` into 4 classes for Cheetah and compare the policies in Table 2. In general, we observe degradation in overall style-consistency accuracies as the number of classes increase. However, CTVAE-style policies still consistently achieve better style-consistency than baselines in this setting.

We visualize and compare policies calibrated for 6 classes of `DISPLACEMENT` in Figure 2. In Figure 2b and 2d, we see that our CTVAE-policy (0.92 style-consistency) is effectively calibrated for styles of low and high displacement, as all trajectories end in the correct corresponding regions (marked by the green dots). On the other hand, trajectories from a baseline CTVAE model (0.70 style-consistency) in Figure 2a and 2c can sometimes end in the wrong region corresponding to a different style label (marked by red dots). These results suggest that incorporating programmatic style-consistency while training via (8) can yield good qualitative and quantitative calibration results.

Model	Speed	Disp.	Dest.	Dir.	Curve
CTVAE	83	72	82	77	61
CTVAE-info	84	71	79	72	60
CTVAE-mi	86	74	82	77	72
CTVAE-style	**95**	**96**	**97**	**97**	**81**

(a) Style-consistency for labeling functions in Basketball.

Model	Speed	Torso	BFoot	FFoot
CTVAE	59	63	68	68
CTVAE-info	57	63	65	66
CTVAE-mi	60	65	65	70
CTVAE-style	**79**	**80**	**80**	**77**

(b) Style-consistency for labeling functions in Cheetah.

Table 1. **Individual Style Calibration:** Style-consistency ($\times 10^{-2}$, median over 5 seeds) of policies evaluated with 4,000 Basketball and 500 Cheetah rollouts. Trained separately for each style, CTVAE-style policies outperform baselines for all styles in Basketball and Cheetah environments.

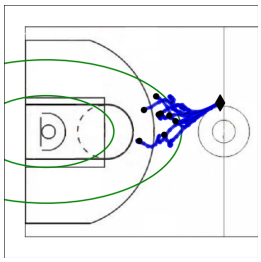
	Basketball				Cheetah	
Model	2 class	3 class	4 class	6 class	3 class	4 class
CTVAE	92	83	79	70	45	37
CTVAE-info	90	83	78	70	49	39
CTVAE-mi	92	84	77	70	48	37
CTVAE-style	**99**	**98**	**96**	**92**	**59**	**51**

Table 2. **Fine-grained Style-consistency:** ($\times 10^{-2}$, median over 5 seeds) Training on labeling functions with more classes (`DISPLACEMENT` for Basketball, `SPEED` for Cheetah) yields increasingly fine-grained calibration of behavior. Although CTVAE-style degrades as the number of classes increases, it outperforms baselines for all styles.

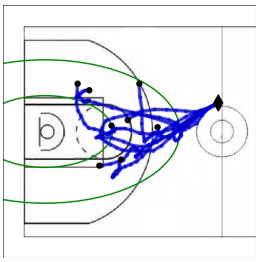
6.2. Can we calibrate for combinatorial joint style spaces?

We now consider combinatorial style-consistency as in (6), which measures the style-consistency with respect to *all* labeling functions simultaneously. For instance, in Figure 3, we calibrate to both terminating close to the net and also the speed at which the agent moves towards the target destination; if either style is not calibrated then the joint style is not calibrated. In our experiments, we evaluated up to 1024 joint styles.

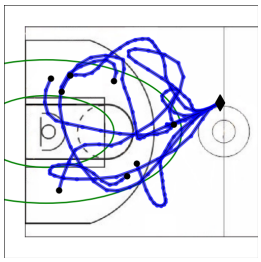
Table 3 compares the style-consistency of policies simultaneously calibrated for up to 5 labeling functions for Basketball and 3 labeling functions for Cheetah. This is a very difficult task, and we see that style-consistency for base-



(a) Label class 0 (slow)



(b) Label class 1 (mid)



(c) Label class 2 (fast)

Figure 3. CTVAE-style rollouts calibrated for 2 styles: label class 1 of `DESTINATION(net)` (see Figure 5 in Appendix D) and each class for `SPEED`, with 0.93 style-consistency. Diamonds (⧫) and dots (●) indicate initial and final positions.

Model	2 style 3 class (8)	3 style 3 class (27)	4 style 3 class (81)	5 style 3 class (243)	5 style 4 class (1024)
CTVAE	71	58	50	37	21
CTVAE-info	69	58	51	32	21
CTVAE-mi	72	56	51	30	21
CTVAE-style	**93**	**88**	**88**	**75**	**55**

(a) Style-consistency for labeling functions in Basketball.

Model	2 style 2 class (4)	3 style 2 class (8)
CTVAE	41	28
CTVAE-info	41	27
CTVAE-mi	40	28
CTVAE-style	**54**	**40**

(b) Style-consistency for labeling functions in Cheetah.

Table 3. **Combinatorial Style-consistency:** ($\times 10^{-2}$, median over 5 seeds) Simultaneously calibrated to joint styles from multiple labeling functions, CTVAE-style policies significantly outperform all baselines. The number of distinct style combinations are in brackets. The most challenging experiment for basketball calibrates for 1024 joint styles (5 labeling functions, 4 classes each), in which CTVAE-style has a $+161\%$ improvement in style-consistency over the best baseline.

lines degrades significantly as the number of joint styles grows combinatorially. On the other hand, our CTVAE-style approach experiences only a modest decrease in style-consistency and is still significantly better calibrated (0.55 style-consistency vs. 0.21 best baseline style-consistency in the most challenging experiment for Basketball). We visualize a CTVAE-style policy calibrated for two styles in Basketball with style-consistency 0.93 in Figure 3. CTVAE-style outperforms baselines in Cheetah as well, but there is still room for improvement to optimize style-consistency better in future work.

6.3. What is the trade-off between style-consistency and imitation quality?

In Table 4, we investigate whether CTVAE-style's superior style-consistency comes at a significant cost to imitation quality, since we optimize both in (5). For Basketball, high style-consistency is achieved without any degradation in imitation quality. For Cheetah, negative log-density is slightly worse; a followup experiment in Table 13 in Appendix D shows that we can improve imitation quality with more training, sometimes with modest decrease to style-consistency.

	Basketball		Cheetah	
Model	D_{KL}	NLD	D_{KL}	NLD
TVAE	2.55	-7.91	29.4	-0.60
CTVAE	2.51	-7.94	29.3	-0.59
CTVAE-info	2.25	-7.91	29.1	-0.58
CTVAE-mi	2.56	-7.94	28.5	-0.57
CTVAE-style	2.27	-7.83	30.1	-0.28

Table 4. KL-divergence and negative log-density per timestep for TVAE models (lower is better). CTVAE-style is comparable to baselines for Basketball, but is slightly worse for Cheetah.

	Style-consistency ↑			
Model	**Min**	**Median**	**Max**	NLD ↓
RNN	79	80	81	**-7.7**
RNN-style	**81**	**91**	**98**	-7.6

Table 5. Style-consistency of RNN policy model (10^{-2}, 5 seeds) for `DESTINATION` in basketball. Our approach improves style-consistency without significantly decreasing imitation quality.

6.4. Is our framework compatible with other policy classes for imitation?

We highlight that our framework introduced in Section 5 is compatible with any policy class. In this experiment, we optimize for style-consistency using a simpler model for the policy and show that style-consistency is still improved. In particular, we use an RNN and calibrate for `DESTINATION` in basketball. In Table 5, we see that style-consistency is improved for the RNN model without any significant decrease in imitation quality.

6.5. What if labeling functions are noisy?

So far, we have demonstrated that our method optimizing for style-consistency directly can learn policies that are much better calibrated to styles, without a significant degradation in imitation quality. However, we note that the labeling functions used thus far are assumed to be perfect, in that they capture exactly the style that we wish to calibrate. In practice, domain experts may specify labeling functions that are noisy; we simulate that scenario in this experiment.

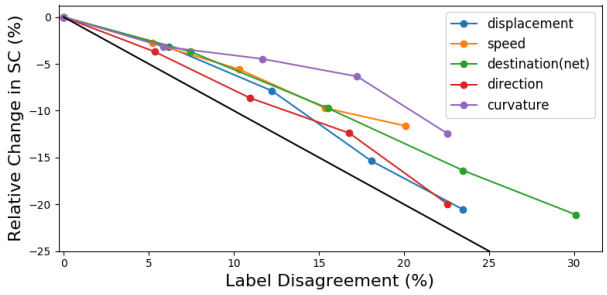


Figure 4. Relative change of style-consistency for CTVAE-style policies trained with noisy labeling functions, which are created by injecting noise with mean 0 and standard deviation $c \cdot \sigma$ for $c \in \{1, 2, 3, 4\}$ before applying thresholds to obtain label classes. The x-axis is the label disagreement between noisy and true labeling functions. The y-axis is the median change (5 seeds) in style-consistency using the true labeling functions without noise, relative to Table 1. The relationship is generally linear and better than a one-to-one dependency (i.e. if $X\%$ label disagreement leads to $-X\%$ relative change, indicated by the black line). See Table 17 and 18 in the Appendix D for more details.

In particular, we create noisy versions of labeling functions in Table 1 by adding Gaussian noise to computed values before applying the thresholds. The noise will result in some label disagreement between noisy and true labeling functions (Table 17 in Appendix D). This resembles the scenario in practice where domain experts can mislabel a trajectory, or have disagreements. We train CTVAE-style models with noisy labeling functions and compute style-consistency using the true labeling functions without noise. Intuitively, we expect the relative decrease in style-consistency to scale linearly with the label disagreement.

Figure 4 shows that the median relative decrease in style-consistency of our CTVAE-models scales linearly with label disagreement. Our method is also somewhat robust to noise, as $X\%$ label disagreement results in better than $X\%$ relative decrease in style-consistency (black line in Figure 4). Directions for future work include combining multiple noisy labeling functions together to improve style-consistency with respect to a "true" labeling function.

7. Conclusion and Future Work

We propose a novel framework for imitating diverse behavior styles while also calibrating to desired styles. Our framework leverages labeling functions to tractably represent styles and introduces programmatic style-consistency, a metric that allows for fair comparison between calibrated policies. Our experiments demonstrate strong empirical calibration results.

We believe that our framework lays the foundation for many directions of future research. First, can one model more complex styles not easily captured with a single labeling function (e.g. aggressive vs. passive play in sports) by composing simpler labeling functions (e.g. max speed, distance to closest opponent, number of fouls committed, etc.), similar to (Ratner et al., 2016; Bach et al., 2017)? Second, can we use these per-timestep labels to model transient styles, or simplify the credit assignment problem when learning to calibrate? Third, can we blend our programmatic supervision with unsupervised learning approaches to arrive at effective semi-supervised solutions? Fourth, can we use model-free approaches to further optimize self-consistency, e.g., to fine-tune from our model-based approach? Finally, can we integrate our framework with reinforcement learning to also optimize for environmental rewards?

Acknowledgements

This research is supported in part by NSF #1564330, NSF #1918655, DARPA PAI, and gifts from Intel, Activision/Blizzard and Northrop Grumman. Basketball dataset was provided by STATS.

References

Bach, S. H., He, B. D., Ratner, A., and Ré, C. Learning the structure of generative models without labeled data. In *International Conference on Machine Learning (ICML)*, 2017.

Bao, J., Chen, D., Wen, F., Li, H., and Hua, G. CVAE-GAN: fine-grained image generation through asymmetric training. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

Bengio, Y., Courville, A. C., and Vincent, P. Unsupervised feature learning and deep learning: A review and new perspectives. *arXiv preprint arXiv:1206.5538*, 2012.

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

Branson, K., Robie, A. A., Bender, J., Perona, P., and Dickinson, M. H. High-throughput ethomics in large groups of drosophila. *Nature methods*, 6(6):451, 2009.

Broll, B., Hausknecht, M., Bignell, D., and Swaminathan, A. Customizing scripted bots: Sample efficient imitation learning for human-like behavior in minecraft. *AAMAS Workshop on Adaptive and Learning Agents*, 2019.

Chang, M.-F., Lambert, J., Sangkloy, P., Singh, J., Bak, S., Hartnett, A., Wang, D., Carr, P., Lucey, S., Ramanan, D., et al. Argoverse: 3d tracking and forecasting with rich maps. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Chen, J., Le, H. M., Carr, P., Yue, Y., and Little, J. J. Learning online smooth predictors for realtime camera planning using recurrent decision trees. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4688–4696, 2016a.

Chen, T. Q., Li, X., Grosse, R. B., and Duvenaud, D. K. Isolating sources of disentanglement in variational autoencoders. In *Neural Information Processing Systems (NeurIPS)*, 2018.

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Neural Information Processing Systems (NeurIPS)*, 2016b.

Cho, K., van Merrienboer, B., Gülçehre, Ç., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Co-Reyes, J. D., Liu, Y., Gupta, A., Eysenbach, B., Abbeel, P., and Levine, S. Self-consistent trajectory autoencoder: Hierarchical reinforcement learning with trajectory embeddings. In *International Conference on Machine Learning (ICML)*, 2018.

Creswell, A., Bharath, A. A., and Sengupta, B. Adversarial information factorization. *arXiv preprint arXiv:1711.05175*, 2017.

Ding, Y., Florensa, C., Abbeel, P., and Phielipp, M. Goal-conditioned imitation learning. In *Neural Information Processing Systems (NeurIPS)*, 2019.

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

Eyjolfsdottir, E., Branson, K., Yue, Y., and Perona, P. Learning recurrent representations for hierarchical behavior modeling. In *International Conference on Learning Representations (ICLR)*, 2017.

Ha, D. and Eck, D. A neural representation of sketch drawings. In *International Conference on Learning Representations (ICLR)*, 2018.

Hausman, K., Chebotar, Y., Schaal, S., Sukhatme, G. S., and Lim, J. J. Multi-modal imitation learning from unstructured demonstrations using generative adversarial nets. In *Neural Information Processing Systems (NeurIPS)*, 2017.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*, 2017.

Ho, J. and Ermon, S. Generative adversarial imitation learning. In *Advances in neural information processing systems*, pp. 4565–4573, 2016.

Hofmann, K. Minecraft as ai playground and laboratory. In *Proceedings of the Annual Symposium on Computer-Human Interaction in Play*, pp. 1–1, 2019.

Hu, Z., Yang, Z., Liang, X., Salakhutdinov, R., and Xing, E. P. Toward controlled generation of text. In *International Conference on Machine Learning (ICML)*, 2017.

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

Kim, H. and Mnih, A. Disentangling by factorising. In *International Conference on Machine Learning (ICML)*, 2018.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

Klys, J., Snell, J., and Zemel, R. S. Learning latent subspaces in variational autoencoders. In *Neural Information Processing Systems (NeurIPS)*, 2018.

Kostrikov, I. Pytorch implementations of reinforcement learning algorithms. https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail, 2018.

Kurin, V., Nowozin, S., Hofmann, K., Beyer, L., and Leibe, B. The atari grand challenge dataset. *arXiv preprint arXiv:1705.10998*, 2017.

Lample, G., Zeghidour, N., Usunier, N., Bordes, A., Denoyer, L., and Ranzato, M. Fader networks: Manipulating images by sliding attributes. In *Neural Information Processing Systems (NeurIPS)*, 2017.

Le, H. M., Carr, P., Yue, Y., and Lucey, P. Data-driven ghosting using deep imitation learning. In *MIT Sloan Sports Analytics Conference (SSAC)*, 2017a.

Le, H. M., Yue, Y., Carr, P., and Lucey, P. Coordinated multi-agent imitation learning. In *International Conference on Machine Learning (ICML)*, 2017b.

Le, H. M., Jiang, N., Agarwal, A., Dudík, M., Yue, Y., and Daumé III, H. Hierarchical imitation and reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2018.

Li, Y., Song, J., and Ermon, S. Infogail: Interpretable imitation learning from visual demonstrations. In *Neural Information Processing Systems (NeurIPS)*, 2017.

Li, Y., Yu, R., Shahabi, C., and Liu, Y. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations (ICLR)*, 2018.

Ling, H. Y., Zinno, F., Cheng, G., and van de Panne, M. Character controllers using motion vaes. In *ACM Conference on Graphics (SIGGRAPH)*, 2020.

Locatello, F., Bauer, S., Lucic, M., Gelly, S., Schölkopf, B., and Bachem, O. Challenging common assumptions in the unsupervised learning of disentangled representations. In *International Conference on Machine Learning (ICML)*, 2019.

Losey, D. P., Srinivasan, K., Mandlekar, A., Garg, A., and Sadigh, D. Controlling assistive robots with learned latent actions. In *International Conference on Robotics and Automation (ICRA)*, 2020.

Luo, W., Yang, B., and Urtasun, R. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Luo, Y., Xu, H., Li, Y., Tian, Y., Darrell, T., and Ma, T. Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees. In *International Conference on Learning Representations (ICLR)*, 2019.

Luo, Y.-S., Soeseno, J. H., Chen, T. P.-C., and Chen, W.-C. Carl: Controllable agent with reinforcement learning for quadruped locomotion. In *ACM Conference on Graphics (SIGGRAPH)*, 2020.

Nagabandi, A., Kahn, G., Fearing, R. S., and Levine, S. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *International Conference on Robotics and Automation (ICRA)*, 2018.

Nishimura, T., Hashimoto, A., Yamakata, Y., and Mori, S. Frame selection for producing recipe with pictures from an execution video of a recipe. In *Proceedings of the 11th Workshop on Multimedia for Cooking and Eating Activities*, pp. 9–16. ACM, 2019.

Ratner, A., Sa, C. D., Wu, S., Selsam, D., and Ré, C. Data programming: Creating large training sets, quickly. In *Neural Information Processing Systems (NeurIPS)*, 2016.

Ross, S., Gordon, G., and Bagnell, D. 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.

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

Sharma, A., Gu, S., Levine, S., Kumar, V., and Hausman, K. Dynamics-aware unsupervised discovery of skills. In *International Conference on Learning Representations (ICLR)*, 2020.

Suwajanakorn, S., Seitz, S. M., and Kemelmacher-Shlizerman, I. Synthesizing obama: learning lip sync from audio. *ACM Transactions on Graphics (TOG)*, 36(4):95, 2017.

Syed, U. and Schapire, R. E. A game-theoretic approach to apprenticeship learning. In *Advances in neural information processing systems*, pp. 1449–1456, 2008.

Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., de Las Casas, D., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., Lillicrap, T. P., and Riedmiller, M. A. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Taylor, S., Kim, T., Yue, Y., Mahler, M., Krahe, J., Rodriguez, A. G., Hodgins, J., and Matthews, I. A deep learning approach for generalized speech animation. *ACM Transactions on Graphics (TOG)*, 36(4):93, 2017.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *International Conference on Intelligent Robots and Systems (IROS)*, 2012.

Wang, Z., Merel, J., Reed, S., Wayne, G., de Freitas, N., and Heess, N. Robust imitation of diverse behaviors. In *Neural Information Processing Systems (NeurIPS)*, 2017.

Yeh, R. A., Schwing, A. G., Huang, J., and Murphy, K. Diverse generation for multi-agent sports games. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Zhan, E., Zheng, S., Yue, Y., Sha, L., and Lucey, P. Generating multi-agent trajectories using programmatic weak supervision. In *International Conference on Learning Representations (ICLR)*, 2019.

Zheng, S., Yue, Y., and Hobbs, J. Generating long-term trajectories using deep hierarchical networks. In *Neural Information Processing Systems (NeurIPS)*, 2016.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 2223–2232, 2017.

A. Baseline Policy Models

1) Conditional-TVAE (CTVAE). The conditional version of TVAEs optimizes:

$$\mathcal{L}^{\text{ctvae}}(\tau, \pi_\theta; q_\phi) = \mathbb{E}_{q_\phi(\mathbf{z}|\tau.\mathbf{y})}\left[\sum_{t=1}^{T} -\log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t, \mathbf{z}, \mathbf{y})\right] + D_{KL}\big(q_\theta(\mathbf{z}|\tau, \mathbf{y})||p(\mathbf{z})\big). \tag{10}$$

2) CTVAE with information factorization (CTVAE-info). (Creswell et al., 2017; Klys et al., 2018) augment conditional-VAE models with an auxiliary network $A_\psi(\mathbf{z})$ which is trained to predict the label $\mathbf{y}$ from $\mathbf{z}$, while the encoder q_ϕ is also trained to minimize the accuracy of A_ψ. This model *implicitly* maximizes self-consistency by removing the information correlated with $\mathbf{y}$ from $\mathbf{z}$, so that any information pertaining to $\mathbf{y}$ that the decoder needs for reconstruction must all come from $\mathbf{y}$. While this model was previously used for image generation, we extend it into the sequential domain:

$$\max_{\theta,\phi}\left(\mathbb{E}_{q_\phi(\mathbf{z}|\tau)}\left[\min_{\psi}\mathcal{L}^{\text{aux}}\big(A_\psi(\mathbf{z}), \mathbf{y}\big) + \sum_{t=1}^{T}\log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t, \mathbf{z}, \mathbf{y})\right] - D_{KL}\big(q_\theta(\mathbf{z}|\tau)||p(\mathbf{z})\big)\right). \tag{11}$$

Note that the encoder in (10) and (11) differ in that $q_\phi(\mathbf{z}|\tau)$ is no longer conditioned on the label $\mathbf{y}$.

3) CTVAE with mutual information maximization (CTVAE-mi). In addition to (10), we can also maximize the mutual information between labels and trajectories $I(\mathbf{y}; \tau)$. This quantity is hard to maximize directly, so instead we maximize the variational lower bound:

$$I(\mathbf{y}; \tau) \geq \mathbb{E}_{\mathbf{y}\sim p(\mathbf{y}), \tau\sim\pi_\theta(\cdot|\mathbf{z},\mathbf{y})}\big[\log r_\psi(\mathbf{y}|\tau)\big] + \mathcal{H}(\mathbf{y}), \tag{12}$$

where r_ψ approximates the true posterior $p(\mathbf{y}|\tau)$. In our setting, the prior over labels is known, so $\mathcal{H}(\mathbf{y})$ is a constant. Thus, the learning objective is:

$$\mathcal{L}^{\text{ctvae-mi}}(\tau, \pi_\theta; q_\phi) = \mathcal{L}^{\text{ctvae}}(\tau, \pi_\theta) + \mathbb{E}_{\mathbf{y}\sim p(\mathbf{y}), \tau\sim\pi_\theta(\cdot|\mathbf{z},\mathbf{y})}\big[-\log r_\psi(\mathbf{y}|\tau)\big]. \tag{13}$$

Optimizing (13) also requires collecting rollouts with the current policy, so similarly we also pretrain and fine-tune a dynamics model M_φ. This baseline can be interpreted as a supervised analogue of unsupervised models that maximize mutual information in (Li et al., 2017; Hausman et al., 2017).

B. Stochastic Dynamics Function

If the dynamics function f of the environment is stochastic, we modify our approach in Algorithm 2 by changing the form of our dynamics model. We can model the change in state as a Gaussian distribution and minimize the negative log-likelihood:

$$\varphi_\mu^*, \varphi_\sigma^* = \arg\min_{\varphi_\mu,\varphi_\mu} \mathbb{E}_{\tau\sim\mathcal{D}} \sum_{t=1}^{T} -\log p(\Delta_t; \mu_t, \sigma_t), \tag{14}$$

where $\Delta_t = \mathbf{s}_{t+1} - \mathbf{s}_t$, $\mu_t = M_{\varphi_\mu}(\mathbf{s}_t, \mathbf{a}_t)$, $\sigma_t = M_{\varphi_\sigma}(\mathbf{s}_t, \mathbf{a}_t)$, and M_{φ_μ}, M_{φ_σ} are neural networks that can share weights. We can sample a change in state during rollouts using the reparametrization trick (Kingma & Welling, 2014), which allows us to backpropagate through the dynamics model during training.

C. Experiment Details

Dataset details. See Table 6. Basketball trajectories are collected from tracking real players in the NBA. Figure 7 shows the distribution of basketball labeling functions applied on the training set. For Cheetah, we train 125 policies using PPO (Schulman et al., 2017) to run forwards at speeds ranging from 0 to 4 (m/s). We collect 25 trajectories per policy by sampling actions from the policy. We use (Kostrikov, 2018) to interface with (Tassa et al., 2018). Figure 8 shows the distributions of Cheetah labeling functions applied on the training set.

Hyperparameters. See Table 7 for training hyperparameters and Table 8 for model hyperparameters.

D. Experiment Results

	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{A}\rvert$	T	N_{train}	N_{test}	frequency (Hz)
Basketball	2	2	24	520,015	67,320	3
Cheetah	18	6	200	2,500	625	40

Table 6. Dataset parameters for basketball and Cheetah environments.

	batch size	# batch b	n_{dynamics}	n_{label}	n_{policy}	n_{collect}	n_{env}	learning rate
Basketball	128	4,063	$10 \cdot b$	$20 \cdot b$	$30 \cdot b$	128	0	$2 \cdot 10^{-4}$
Cheetah	16	157	$50 \cdot b$	$20 \cdot b$	$60 \cdot b$	16	1	10^{-3}

Table 7. Hyperparameters for Algorithm 2. b is the number of batches to see all trajectories in the dataset once. We also use L_2 regularization of 10^{-5} for training the dynamics model M_φ.

	$\mathbf{z}$-dim	q_ϕ GRU	C_ψ^λ GRU	π_θ GRU	π_θ sizes	M_φ sizes
Basketball	4	128	128	128	(128,128)	(128,128)
Cheetah	8	200	200	-	(200,200)	(500,500)

Table 8. Model parameters for basketball and Cheetah environments.

Model	**Speed**			**Displacement**			**Destination**			**Direction**			**Curvature**		
CTVAE	82	83	85	71	72	74	81	82	82	76	77	80	60	61	62
CTVAE-info	**84**	84	87	69	71	74	78	79	83	71	72	74	60	60	62
CTVAE-mi	84	86	87	71	74	74	80	82	84	75	77	78	58	72	74
CTVAE-style	34	**95**	**97**	**89**	**96**	**97**	**91**	**97**	**98**	**96**	**97**	**98**	**77**	**81**	**83**

(a) Style-consistency wrt. single styles of 3 classes (roughly uniform distributions).

Model	**2 classes**			**3 classes**			**4 classes**			**6 classes**			**8 classes**		
CTVAE	91	92	93	79	83	84	**76**	79	79	**68**	70	72	64	66	69
CTVAE-info	90	90	92	**83**	83	85	75	76	77	68	70	72	60	63	67
CTVAE-mi	90	92	93	81	84	86	75	77	80	66	70	72	62	62	67
CTVAE-style	**98**	**99**	**99**	15	**98**	**99**	15	**96**	**96**	02	**92**	**94**	**80**	**90**	**93**

(b) Style-consistency wrt. `DISPLACEMENT` of up to 8 classes (roughly uniform distributions).

Model	**2 classes**			**3 classes**			**4 classes**			**6 classes**		
CTVAE	86	87	87	80	82	83	**76**	78	79	70	74	77
CTVAE-info	83	87	88	79	81	83	73	75	78	71	77	78
CTVAE-mi	86	88	88	**80**	81	84	71	74	79	**73**	76	78
CTVAE-style	**97**	**98**	**99**	68	**97**	**98**	35	**89**	**95**	67	**84**	**93**

(c) Style-consistency wrt. `DESTINATION(net)` with up to 6 classes (non-uniform distributions).

Model	**2 styles 3 classes**			**3 styles 3 classes**			**4 styles 3 classes**			**5 styles 3 classes**			**5 styles 4 classes**		
CTVAE	67	71	73	58	58	62	49	50	52	27	37	35	**20**	21	22
CTVAE-info	68	69	70	54	58	59	48	51	54	28	32	35	18	21	23
CTVAE-mi	71	72	73	48	56	61	45	51	52	16	30	31	18	21	23
CTVAE-style	**92**	**93**	**94**	**86**	**88**	**90**	**62**	**88**	**88**	**66**	**75**	**80**	11	**55**	**77**

(d) Style-consistency wrt. multiple styles simultaneously.

Table 9. [min, median, max] style-consistency ($\times 10^{-2}$, 5 seeds) of policies evaluated with 4,000 basketball rollouts each. CTVAE-style policies significantly outperform baselines in all experiments and are calibrated at almost maximal style-consistency for 4/5 labeling functions. We note some rare failure cases with our approach, which we leave as a direction for improvement for future work.

Model	Speed			Torso Height			B-Foot Height			F-Foot Height		
CTVAE	53	59	62	62	63	70	61	68	73	63	68	72
CTVAE-info	56	57	61	62	63	72	58	65	72	63	66	69
CTVAE-mi	53	60	62	62	65	70	60	65	70	66	70	73
CTVAE-style	**68**	**79**	**81**	**79**	**80**	**84**	**77**	**80**	**88**	**74**	**77**	**80**

(a) Style-consistency wrt. single styles of 2 classes (roughly uniform distributions).

Model	3 classes			4 classes		
CTVAE	41	45	49	35	37	41
CTVAE-info	47	49	52	36	39	42
CTVAE-mi	47	48	53	36	37	38
CTVAE-style	**59**	**59**	**65**	**42**	**51**	**60**

(b) Style-consistency wrt. `SPEED` with varying # of classes (non-uniform distributions).

Model	2 styles 2 classe			3 styles 2 classes		
CTVAE	39	41	43	25	28	29
CTVAE-info	39	41	46	25	27	30
CTVAE-mi	34	40	48	27	28	31
CTVAE-style	**43**	**54**	**60**	**38**	**40**	**52**

(c) Style-consistency wrt. multiple styles simultaneously.

Table 10. [min, median, max] style-consistency ($\times 10^{-2}$, 5 seeds) of policies evaluated with 500 Cheetah rollouts each. CTVAE-style policies consistently outperform all baselines, but we note that there is still room for improvement (to reach 100% style-consistency).

Model	Speed	Displacement	Destination	Direction	Curvature
CTVAE	83.4 ± 1.2	72.4 ± **1.4**	81.9 ± **0.6**	77.7 ± 1.3	61.0 ± 1.0
CTVAE-info	85.0 ± **1.2**	71.2 ± 1.9	80.1 ± 1.8	72.3 ± 1.1	60.2 ± **0.8**
CTVAE-mi	**85.8** ± 1.3	72.8 ± 1.5	82.2 ± 1.4	76.9 ± 1.1	68.6 ± 6.4
CTVAE-style	72.1 ± 33.3	**94.6** ± 3.1	**95.0** ± 3.7	**96.8** ± **0.7**	**79.6** ± 2.7

(a) Style-consistency wrt. single styles of 3 classes (roughly uniform distributions).

Model	2 classes	3 classes	4 classes	6 classes	8 classes
CTVAE	92.1 ± 0.9	82.4 ± 2.4	78.0 ± 1.4	69.9 ± **1.4**	66.0 ± **2.0**
CTVAE-info	90.5 ± 0.9	**83.6** ± **1.0**	75.9 ± **0.9**	**70.2** ± 1.6	63.4 ± 2.9
CTVAE-mi	91.6 ± 1.2	83.5 ± 2.1	77.6 ± 2.5	68.8 ± 2.5	63.7 ± 2.3
CTVAE-style	**98.7** ± **0.4**	81.4 ± 36.9	**79.3** ± 35.9	68.1 ± 40.0	**88.2** ± 5.1

(b) Style-consistency wrt. `DISPLACEMENT` of up to 8 classes (non-uniform distributions).

Model	2 classes	3 classes	4 classes	6 classes
CTVAE	86.6 ± **0.6**	81.6 ± **1.3**	**77.4** ± **1.5**	74.0 ± 2.6
CTVAE-info	86.2 ± 1.7	81.1 ± 1.4	75.3 ± 2.5	75.3 ± 3.3
CTVAE-mi	87.3 ± 0.9	81.6 ± 1.6	74.3 ± 3.1	75.8 ± **2.1**
CTVAE-style	**98.1** ± 0.8	**88.2** ± 13.6	77.0 ± 24.1	**82.6** ± 11.3

(c) Style-consistency wrt. `DESTINATION(net)` of up to 6 classes (non-uniform distributions).

Model	2 styles 3 classes	3 styles 3 classes	4 styles 3 classes	5 styles 3 classes	5 styles 4 classes
CTVAE	70.5 ± 2.1	58.9 ± **1.5**	50.4 ± **1.4**	31.6 ± 2.8	**20.8** ± 1.0
CTVAE-info	69.0 ± 0.9	57.5 ± 2.0	50.5 ± 2.3	31.4 ± **2.5**	20.6 ± 2.0
CTVAE-mi	71.8 ± **0.7**	53.8 ± 5.9	50.2 ± 2.7	26.9 ± 6.3	20.7 ± 1.9
CTVAE-style	**92.8** ± 1.0	**88.3** ± 1.7	**81.7** ± 11.0	**73.9** ± 5.4	**50.3** ± 24.7

(d) Style-consistency wrt. multiple styles simultaneously.

Table 11. Mean and standard deviation style-consistency ($\times 10^{-2}$, 5 seeds) of policies evaluated with 4,000 basketball rollouts each. CTVAE-style policies generally outperform baselines. Lower mean style-consistency (and large standard deviation) for CTVAE-style is often due to failure cases, as can be seen from the minimum style-consistency values we report in Table 9. Understanding the causes of these failure cases and improving the algorithm's stability are possible directions for future work.

Model	Speed	Torso Height	B-Foot Height	F-Foot Height
CTVAE	57.4 ± 3.9	64.4 ± 3.1	67.4 ± 4.2	68.5 ± 3.7
CTVAE-info	58.3 ± **2.1**	65.0 ± 4.2	64.1 ± 5.4	66.1 ± 2.7
CTVAE-mi	58.4 ± 3.9	65.7 ± 3.2	65.0 ± **3.6**	69.9 ± 2.6
CTVAE-style	**77.0** ± 5.3	**81.0** ± **2.2**	**81.9** ± 5.4	**77.2** ± **2.4**

(a) Style-consistency wrt. single styles of 2 classes (roughly uniform distributions).

Model	3 classes	4 classes
CTVAE	45.2 ± 3.2	37.8 ± 2.9
CTVAE-info	49.2 ± **1.8**	39.3 ± 2.8
CTVAE-mi	49.1 ± 2.2	36.8 ± **1.0**
CTVAE-style	**60.8** ± 2.9	**51.3** ± 7.8

(b) Style-consistency wrt. `SPEED` with varying # of classes (non-uniform distributions).

Model	2 styles 2 classes	3 styles 2 classes
CTVAE	40.9 ± **1.6**	27.2 ± 1.9
CTVAE-info	41.8 ± 2.3	27.8 ± 2.2
CTVAE-mi	40.7 ± 4.9	28.5 ± **1.6**
CTVAE-style	**52.6** ± 6.1	**42.8** ± 5.8

(c) Style-consistency wrt. multiple styles simultaneously.

Table 12. Mean and standard deviation style-consistency ($\times 10^{-2}$, 5 seeds) of policies evaluated with 500 Cheetah rollouts each. CTVAE-style policies consistently outperform all baselines, but we note that there is still room for improvement (to reach 100% style-consistency).

	Speed		Torso Height		B-Foot Height		F-Foot Height	
Model	NLD	SC	NLD	SC	NLD	SC	NLD	SC
CTVAE-style	-0.28	**79**	-0.24	80	-0.16	**80**	-0.22	**77**
CTVAE-style+	-0.49	70	-0.42	**83**	-0.36	**80**	-0.42	74

Table 13. We report the median negative log-density per timestep (lower is better) and style-consistency (higher is better) of CTVAE-style policies for Cheetah (5 seeds). The first row corresponds to experiments in Tables 1 and 10a, and the second row corresponds to the same experiments with 50% more training iterations. The KL-divergence in the two sets of experiments are roughly the same. Although imitation quality improves, style-consistency can sometimes degrade (e.g. `SPEED`, `FRONT-FOOT HEIGHT`), indicating a possible trade-off between imitation quality and style-consistency.

	Style-consistency ↑					
Model	**Min**	-	**Median**	-	**Max**	NLD ↓
RNN	79	79	80	81	81	-7.7
RNN-style	81	86	91	95	98	-7.6
CTVAE	81	82	82	82	82	**-8.0**
CTVAE-style	**91**	**92**	**97**	**98**	**98**	-7.8

Table 14. Comparing style-consistency ($\times 10^{-2}$) between RNN and CTVAE policy models for `DESTINATION` in basketball. The style-consistency for 5 seeds are listed in increasing order. Our algorithm improves style-consistency for both policy models at the cost of a slight degradation in imitation quality. In general, CTVAE performs better than RNN in both style-consistency and imitation quality.

	Speed	Displacement	Destination	Direction	Curvature
$\mathcal{L}^{\text{label}}$	3.96 ± 0.33	4.58 ± 0.20	1.61 ± 0.18	3.19 ± 0.25	28.31 ± 0.95

(a) Basketball labeling functions for experiments in section 6.1.

	Speed	Torso Height	B-Foot Height	F-Foot Height
$\mathcal{L}^{\text{label}}$	3.24 ± 0.83	15.87 ± 1.78	17.25 ± 0.73	14.75 ± 0.74

(b) Cheetah labeling functions for experiments in section 6.1.

Table 15. Mean and standard deviation cross-entropy loss ($\mathcal{L}^{\text{label}}$, $\times 10^{-2}$) over 5 seeds of learned label approximators $C^{\lambda}_{\psi^*}$ on test trajectories after n^{label} training iterations for experiments in section 6.1. $C^{\lambda}_{\psi^*}$ is only used during training; when computing style-consistency for our quantitative results, we use original labeling functions λ.

	M_φ test error
Basketball	$1.47 \pm 0.59 (\times 10^{-7})$
Cheetah	$1.93 \pm 0.08 (\times 10^{-2})$

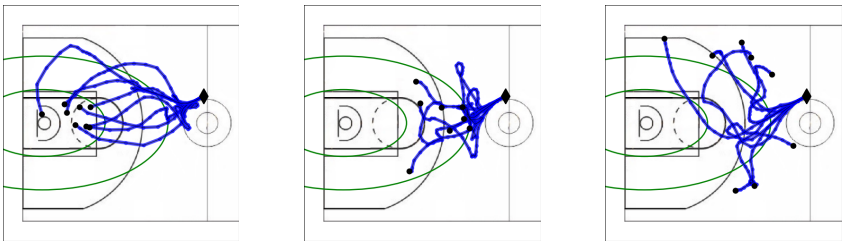
Table 16. Average mean-squared error of dynamics model M_φ per timestep per dimension on test data after training for n^{dynamics} iterations

	Basketball				
noise	**Speed**	**Disp.**	**Dest.**	**Dir.**	**Curve**
σ	5.20	6.18	7.46	5.36	5.88
2σ	10.33	12.24	15.54	10.93	11.66
3σ	15.36	18.08	23.46	16.78	17.24
4σ	20.10	23.47	30.10	22.56	22.52
σ value	0.001	0.02	0.02	0.1	0.02

Table 17. **Label disagreement (%) of noisy labeling functions:** For each of the Basketball labeling functions with 3 classes in Table 1, we consider noisy versions where we inject Gaussian noise with mean 0 and standard deviation $c \cdot \sigma$ for $c \in \{1, 2, 3, 4\}$ before applying thresholds to obtain label classes. This table shows the label disagreement between noisy and true labeling functions over trajectories in the training set. The last row shows the σ value used for each labeling function.

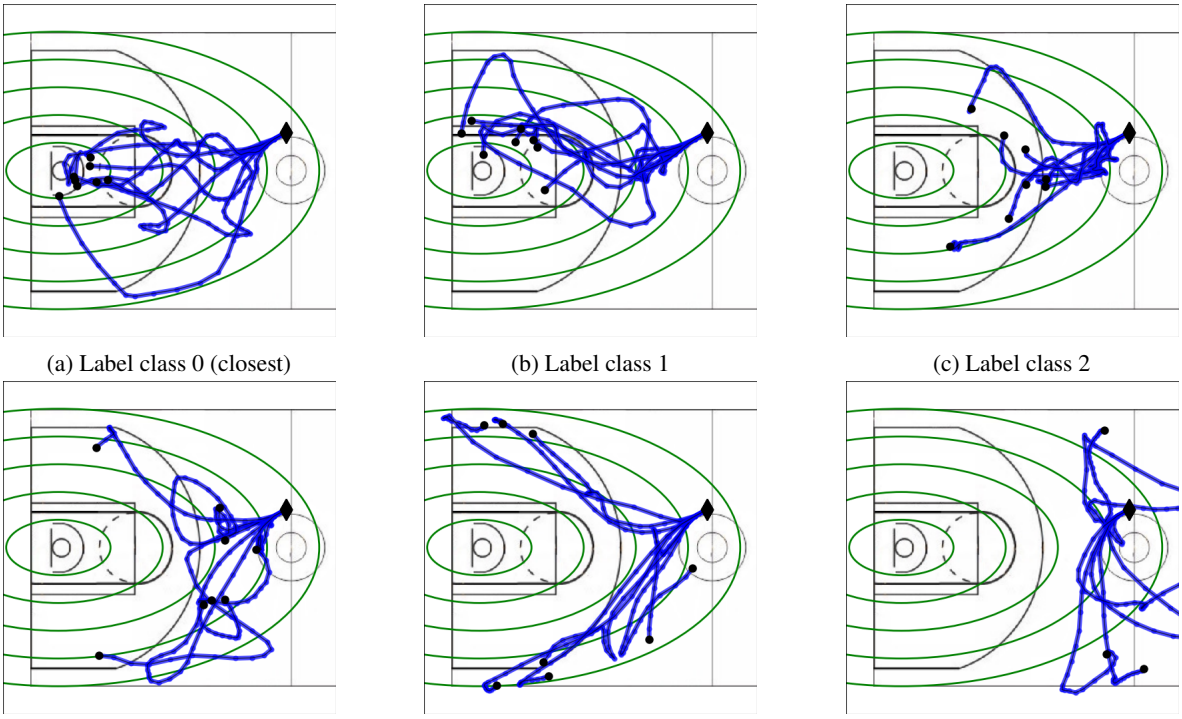
	Basketball				
noise	**Speed**	**Disp.**	**Dest.**	**Dir.**	**Curve**
σ	2.78	3.21	3.70	3.71	3.16
2σ	5.59	7.88	9.75	8.63	4.46
3σ	9.71	15.37	16.38	12.39	6.34
4σ	11.63	20.54	21.11	19.98	12.41

Table 18. **Relative *decrease* in style-consistency** when training with noisy labeling functions: (%, median over 5 seeds) Using the noisy labeling functions in Table 17, we train CTVAE-style models and evaluate style-consistency using the true labeling functions without noise. This table shows the percentage decrease in style-consistency relative to when there is no noise in Table 1. Comparing with the label disagreement in Table 17, we see that the relative decrease in style-consistency generally scales linearly with the label disagreement between noisy and true labeling functions.



(a) Label class 0 (close) (b) Label class 1 (mid) (c) Label class 2 (far)

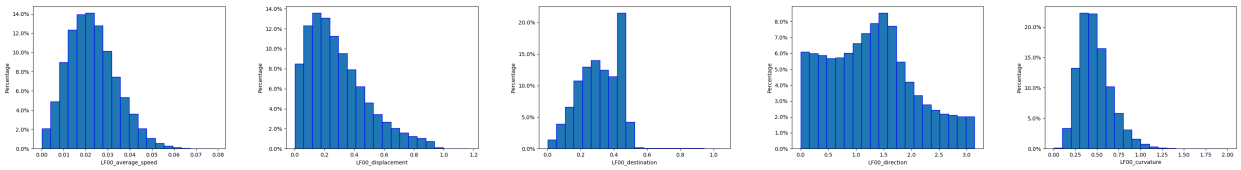
Figure 5. CTVAE-style rollouts calibrated for `DESTINATION(net)`, 0.97 style-consistency. Diamonds (⧫) and dots (•) indicate initial and final positions. Regions divided by green lines represent label classes.



(a) Label class 0 (closest) (b) Label class 1 (c) Label class 2

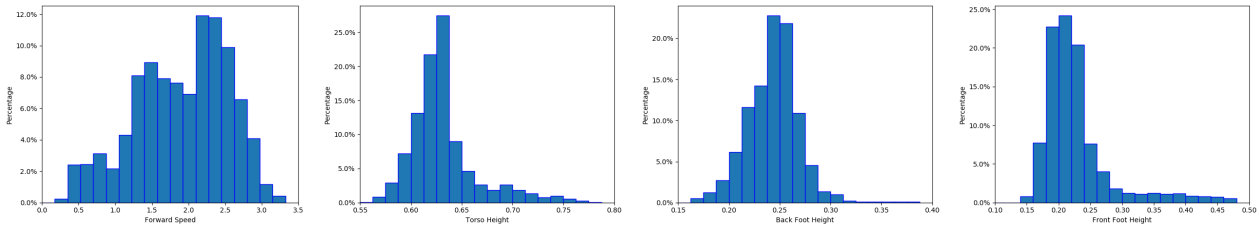
(d) Label class 3 (e) Label class 4 (f) Label class 5 (farthest)

Figure 6. Rollouts from our policy calibrated to `DESTINATION(net)` with 6 classes. The 5 green boundaries divide the court into 6 regions, each corresponding to a label class. The label indicates the target region of a trajectory's final position (●). This policy achieves a style-consistency of 0.93, as indicated in Table 9c. Note that the initial position (⧫) is the same as in Figures 5 and 3 for comparison, but in general we sample an initial position from the prior $p(\mathbf{y})$ to compute style-consistency.



(a) Speed (b) Displacement (c) Destination (d) Direction (e) Curvature

Figure 7. Histogram of basketball labeling functions applied on the training set (before applying thresholds). Basketball trajectories are collected from tracking real players in the NBA.



(a) Speed (b) Torso Height (c) Back-Foot Height (d) Front-Foot Height

Figure 8. Histogram of Cheetah labeling functions applied on the training set (before applying thresholds). Note that `SPEED` is the most diverse behavior because we pre-trained the policies to achieve various speeds when collecting demonstrations, similar to (Wang et al., 2017). For more diversity with respect to other behaviors, we can also incorporate a target behavior as part of the reward when pre-training Cheetah policies.