Generative Hybrid Representations for Activity Forecasting with No-Regret Learning

Jiaqi Guan∗1,2, Ye Yuan1, Kris M. Kitani1, and Nicholas Rhinehart∗1,3

1Carnegie Mellon University 2University of Illinois Urbana Champaign 3UC Berkeley
{jiaqig, yyuan2, kkitani}@cs.cmu.edu, nrhinehart@berkeley.edu

Abstract

Automatically reasoning about future human behaviors is a difficult problem but has significant practical applications to assistive systems. Part of this difficulty stems from learning systems' inability to represent all kinds of behaviors. Some behaviors, such as motion, are best described with continuous representations, whereas others, such as picking up a cup, are best described with discrete representations. Furthermore, human behavior is generally not fixed: people can change their habits and routines. This suggests these systems must be able to learn and adapt continuously. In this work, we develop an efficient deep generative model to jointly forecast a person’s future discrete actions and continuous motions. On a large-scale egocentric dataset, EPIC-KITCHENS, we observe our method generates high-quality and diverse samples while exhibiting better generalization than related generative models. Finally, we propose a variant to continually learn our model from streaming data, observe its practical effectiveness, and theoretically justify its learning efficiency.

1. Introduction

A key requirement for intelligent systems to safely interact with humans is the ability to predict plausible human behaviors. Additionally, they must be able to adapt to variability in behavior over time. However, forecasting a person’s behaviors is generally difficult due to the immense set of possible behaviors that humans showcase. This makes it challenging to choose a unified representation for human behavior. Some behaviors are better modeled as continuous representations, for instance, a person’s future trajectory. Other behaviors are more succinctly represented discretely, such as interacting with an object. Our goal is to develop an efficient predictive model for joint discrete-continuous spaces, which takes rich sensory information from egocentric videos as input to forecast a person’s future behaviors.

For many applications based on a predictive model of future human behavior, it is important that the model is able to characterize the uncertainty of its predictions. A generative model can naturally represent uncertainty and is also well-suited for modeling a hybrid representation of human behavior. Thus, we propose a generative model that can represent the joint distribution of discrete and continuous behaviors by leveraging recent success in generative modeling. Unlike some popular generative models (e.g. GANs [12] and variational autoencoders [21]), our method can compute exact likelihoods, which makes it possible to precisely evaluate the model’s predictions of future behaviors. It is part of a family of methods known as invertible generative models [5, 13, 20]. We learn a generative model of discrete actions by applying the Gumbel-Softmax trick [29], and condition this model on continuous samples produced by an invertible generative trajectory model [36]. We show how we can jointly learn both models efficiently. The results on a large-scale egocentric dataset, EPIC-KITCHENS [4], demonstrate the advantage of our model in joint trajectory-
action forecasting over other generative models and discriminative models. To enable our model to learn optimally from streaming data we employ online learning theories [43]. In particular, we apply a modified objective to fine-tune a subset of the model’s parameters using a no-regret online learning algorithm. We prove our method’s effectiveness theoretically, and observe its online performance matches these theoretical expectations. Example predictions of our method are shown in 1.

We present the following contributions:

1. **Generative hybrid representations**: We propose a generative approach to egocentric forecasting that jointly models trajectory and action distributions. Our experiments on the EPIC-KITCHENS dataset show that our method outperforms both discriminative and generative baselines.

2. **Exact learning and evaluation**: Our model can compute the probability density function (PDF) exactly and also enables optimization of model sample-based metrics (e.g., reverse cross entropy), which renders learning and inference of people’s future trajectory and action more efficient.

3. **Theoretically justified no-regret online fine-tuning**: We extend our model to learn online with a simple, yet effective fine-tuning process. We demonstrate that it is theoretically efficient, which enables the model to learn from data that arrives continuously and the average regret will approach to zero with time elapsing.

### 2. Related Work

We propose a generative model to jointly forecast future trajectories and actions under the first-person vision setting. We begin by discussing work related to our data domain, task, and model.

**First-person vision**: As wearable cameras become more accessible in our daily lives, a growing body of work is using them for understanding human behaviors [7, 26, 42, 27, 32, 51]. The rich visual information encoded in first-person videos can also be used to predict the subject’s attention [27, 55] and their interactions with the environment.

**Trajectory Forecasting**: Third-person trajectory forecasting has enjoyed significant research attention recently. The approach in [25] predicts future trajectories of wide-receivers from surveillance video. A large body of work has also used surveillance video to predict future pedestrian trajectories [49, 28, 2, 22]. Deterministic trajectory modeling has been used for vehicle [17] and pedestrian [1, 37, 50] trajectory prediction. Due to the uncertain nature of future trajectories, modeling stochasticity can help explain multiple plausible trajectories with the same initial context. Several approaches have tried to forecast distributions over trajectories [24, 9]. [36] proposed a generative approach to model vehicle trajectories. A relative small amount of work has investigated trajectory forecasting from first-person videos. [44] predicts the future trajectories of the camera wearer by constructing an EgoRetinal map.

These approaches employed continuous representations in the batch learning setting, while our model uses both discrete and continuous representations in both the batch and online learning settings.

**Action Forecasting**: Classification-based approaches [16, 23, 41, 40] are popular in action forecasting. Many activities are best represented as categories. [10] proposed an encoder-decoder LSTM model to predict future actions. Other work has also tried to forecast more generalized action such as gaze [55], user-object interactions [8] and the position of hands and objects [6]. In [35], online inverse reinforcement learning (IRL) is used to model a person’s goals and future trajectories. IRL has also been applied to forecast the behaviors of robots [33], taxis [57], and pedestrians [22]. Some work has investigated non-discriminative modeling of future actions. [45] devised a deep multi-modal regressor to allow multiple future predictions. [6] uses a variational autoencoder (VAE) to model the distribution of possible future actions. Whereas prior activity forecasting approaches reason about actions only, our method reasons jointly about actions and trajectories.

**Generative Models**: Deep generative models, e.g. [12, 21], are a powerful unsupervised modeling approach. To enable efficient learning of deep generative models of categorical distributions, [29] proposed the Gumbel-Softmax trick to backpropagate gradients through these distributions. There has been work that uses generative models to address the uncertainty in both trajectory [24, 9, 36, 52, 48, 54] and action forecasting [6, 45]. Unlike the prior approaches, our method jointly generates the future trajectories and actions.

**Online Learning**: The field of online learning studies how to learn effectively from streaming data [43], but these approaches are rarely used in computer vision problems. In [35], online inverse reinforcement learning is performed with visual data. In contrast, our approach is based on imitation learning without reward modeling. In [39, 38], interactive imitation learning is framed as a online learning problem. Our approach, while a form of imitation learning, is not interactive. It observes expert behavior (human behaviors) and makes predictions that the human does not interact with.

### 3. Generative Hybrid Activity Forecasting

#### 3.1. Problem Formulation

Our goal is to model the true joint distribution \( p(x, a|\phi) \) of a person’s future trajectory \( x \in \mathbb{R}^{T \times 3} \) in 3D and actions \( a \in \{0, 1\}^{T \times C_x \times 2} \) from egocentric videos with a learned joint distribution \( q(x, a|\phi) \), where \( \phi \) is the context informa-
tion. \( T \) is the forecasting horizon, and \( C_a \) is the number of action classes (with each class modeled with 2 values using a one-hot encoding). The context information \( \phi \) includes past egocentric video frames \( V_{-p,0} \) and positions \( x_{-p,0} \), where \( P \) is the observation horizon.

As \( x \) and \( a \) use different representations (continuous vs. discrete), we further factorize the joint distribution by conditioning \( a \) on \( x \) i.e. \( q(x, a|\phi) = q(x|\phi)q(a|x, \phi) \). Learning this model of future behavior via divergence minimization is akin to imitation learning \([11, 19]\). We use one-step policies \( \pi \) for generating trajectory \( x \) and \( \kappa \) for generating actions \( a \), and samples from \( q_{\pi}(x|\phi) \) and \( q_{\kappa}(a|x, \phi) \) can be obtained by repeatedly sampling \( T \) times from \( \pi \) and \( \kappa \). These policies parameterize each generative model. Our training data is a set of episodes denoted \( \{(x, a, \phi)\}_{n=1}^{N} \), which are samples from the (unknown) data distribution of the person’s behavior \( p(x, a|\phi) \). We use this data to train the policies \( \pi \) and \( \kappa \), thereby learning \( q(x, a|\phi) \).

### 3.2. Complementary Cross Entropy Loss

A desired feature of forecasting models is to generate both diverse and precise predictions. Following \([36]\), we construct a complementary cross-entropy loss to train our trajectory-action distribution \( q(x, a|\phi) \):

\[
\mathcal{L} = \frac{1}{H(p,q)} \mathbb{E}_{(x,\phi) \sim p} - \log q(x, a|\phi) + \beta \frac{1}{H(q, \tilde{p})} \mathbb{E}_{(x,\phi) \sim q} - \log \tilde{p}(x, a|\phi),
\]

where \( \tilde{p} \) is an approximation to the data distribution \( p \), which we will discuss in detail in Sec 3.6. \( \beta \) is a weighting factor. The forward cross entropy term \( H(p,q) \) encourages the distribution \( q \) to cover all modes of \( p \) and thus increases sample diversity. The reverse cross entropy term \( H(q, \tilde{p}) \) penalizes samples far from the data distribution \( \tilde{p} \) to improve sample quality. The joint use of them promotes both diversity and quality of samples. We use \( \beta \) to control the trade-off between diversity and precision.

With the factorization \( q(x, a|\phi) = q_{\pi}(x|\phi)q_{\kappa}(a|x, \phi) \), the forward and reverse cross entropies can be rewritten as

\[
H(p, q) = \mathbb{E}_{x \sim p} \log q_{\pi}(x|\phi) - \mathbb{E}_{(x, a) \sim p} \log q_{\kappa}(a|x, \phi),
\]

\[
H(q, \tilde{p}) = \mathbb{E}_{x \sim q} \log \tilde{p}(x|\phi) - \mathbb{E}_{(x, a) \sim q} \log \tilde{p}(a|x, \phi).
\]

This decomposition disentangles the cross entropies for trajectory and actions, allowing us to learn the policy \( \pi \) and \( \kappa \) separately. The optimization of \( H(p, q) \) requires us to compute \( q \) and the optimization of \( H(q, \tilde{p}) \) requires us to sample from \( q \). Different from GANs \([12]\) (likelihood-free learning) and VAEs \([21]\) (optimize the evidence lower bound), we propose an invertible generative model, which enables us to both compute the likelihood of \( q(x, a|\phi) \) exactly and generate samples from \( q(x, a|\phi) \). The model details will be illustrated in Sec 3.3, 3.4 and 3.5.

### 3.3. Trajectory Cross Entropy

We employ an invertible trajectory generative model by constructing a differentiable, invertible function \( f_{\pi}(z; \phi) : \mathbb{R}^{T \times 3} \rightarrow \mathbb{R}^{T \times 3} \). This function maps a noise sequence \( z = [z_1, \ldots, z_T] \) from a Gaussian distribution \( \mathcal{N}(0, I_{3 \times 3}) \) and the scene context \( \phi \) to a trajectory \( x = [x_1, \ldots, x_T] \). \( f_{\pi} \) is implemented by a \( \theta \)-parametrized per-step policy \( \pi \). At each time step \( t \), \( \pi \) takes in a per-step context \( \psi_t \), containing past positions \( x_{t-\tau:t-1} \), and outputs the mean \( \mu_t \) and an invertible covariance matrix \( \sigma_t \), and simulate the current position \( x_t \) with noise \( z_t \): \( x_t \triangleq \mu_t (\psi_t; \theta) + \sigma_t (\psi_t; \theta) z_t \). Since \( \sigma_t \) is invertible, \( \pi \) defines a bijection between \( z_t \) and
\[ q_\pi \text{ then follows from the change-of-variables formula for multivariate integration} \]  
\[ q_\pi (x|\phi) = \mathcal{N} (f_\pi^{-1}(x; \phi)) |\det \partial f_\pi^{-1}(x; \phi)|^{-1}, \]  
(3)

where \( J_{f_\pi}(f_\pi^{-1}(x; \phi)) \) is the Jacobian of \( f_\pi \) evaluated at \( f_\pi^{-1}(x; \phi) \). Thus, the forward cross entropy can be rewritten as

\[ H(p, q_\pi) = - \mathbb{E}_{x \sim \tilde{p}} \log \frac{\mathcal{N}(f_\pi^{-1}(x; \phi))}{|\det \partial f_\pi^{-1}(x; \phi)|}. \]  
(4)

The reparameterization also greatly simplifies the differentiation of \( H(q_\pi, \tilde{p}) \) w.r.t. policy \( \pi \). Instead of sampling from \( q_\pi \), we can sample from \( \mathcal{N} \) and rewrite the reverse cross entropy as Eq. (5). \( z \) is the source of uncertainty for generating diverse samples.

\[ H(q_\pi, \tilde{p}) = - \mathbb{E}_{z \sim \mathcal{N}} \log \tilde{p}(f_\pi(z; \phi)) \]  
(5)

### 3.4. Action Cross Entropy

For the action forecasting, at each step \( t \) each action class \( c \) is represented as \( a_{t,c} \in \{0, 1\}^2 \) which is a one-hot vector indicating whether this action happens \((0, 1)\) or not \((1, 0)\). Since actions are discrete variables, we use Gumbel-Softmax distributions [18] to reparameterize actions. We build a simulator \( h_\kappa (g; \phi) : \mathbb{R}^{T \times C_t \times 2} \rightarrow \{0, 1\}^{T \times C_t \times 2} \), which maps noise sequences \( g \) sampled from Gumbel distribution \( \mathcal{G}(0, 1) \) to actions \( a \). The noise sequence \( g \), as a key part of the Gumbel-Softmax reparameterization – a continuous, differentiable approximation to Gumbel-Max, provides an efficient way to draw samples from a categorical distribution.

The per-step action forecasting context \( \chi_t \) consists of past images \( V_{t-p,t-1} \) and past positions \( x_{t-p,t-1} \). The per-step policy \( \kappa \) outputs action probabilities \( u_t \) with \( \chi_t \), and simulate the current action \( a_t \) with noise \( g_t \):

\[ a_{t,c,i} \triangleq \frac{\exp ((u_{t,c,i}(\chi_t; \theta) + g_{t,c,i})/\tau)}{\sum_{j \in \{1,2\}} \exp ((u_{t,c,j}(\chi_t; \theta) + g_{t,c,j})/\tau)}, \]

where \( i \in \{1,2\}, c \in \{1, \ldots, C_t\}, \) and \( t \in \{1, \ldots, T\} \), \( \tau \) is the temperature of Gumbel-Softmax distribution.

According to the probability density function of the Gumbel-Softmax distribution [18], the action forward cross entropy can be rewritten as

\[ H(p, q_\kappa) = - \mathbb{E}_{(x,a) \sim p} \log \left( \sum_{i=1}^2 \frac{u_{t,c,i}(\chi_t)}{a_{t,c,i}} \right)^{-2} \prod_{i=1}^2 \frac{u_{t,c,i}(\chi_t)}{a_{t,c,i}^{1/2}}. \]  
(6)

For the reverse cross entropy, using Gumbel-Softmax reparameterization, it can be rewritten as

\[ H(q_\kappa, \tilde{p}) = - \mathbb{E}_{a \sim \tilde{p}} \sum_{t,c,i} \log \tilde{p}(a_{t,c,i}|x, \phi). \]  
(7)

The overall procedure of training the batch model is shown in Algorithm 1.

---

**Algorithm 1** Offline Generative Hybrid Activity Forecasting

**Require:** Training dataset \( \{(x, a, \phi)_n\}_{n=1}^N \); Batch size \( B \);

- Trajectory simulator \( f_\pi \); Action simulator \( h_\kappa \);
- Pre-trained weights \( \theta \)

1. Randomly initialize \( f_\pi \) and \( h_\kappa \) with parameter \( \theta \)
2. repeat
   3. for each mini-batch examples \( (x, a, \phi)_{i:t+B} \) do
      4. Calculate \( H(p, q_\pi) \) with Eq. (4) (6)
      5. Sample \( z \sim \mathcal{N} \); Generate trajectory \( \hat{x} = f_\pi(z; \phi) \)
      6. Calculate \( H(q_\pi, \tilde{p}) \) with Eq. (5)
      7. Sample \( g \sim \mathcal{G} \); Generate actions \( \hat{a} = h_\kappa(g; \phi) \)
      8. Calculate \( H(q_\kappa, \tilde{p}) \) with Eq. (7)
   9. Update \( \theta \) by optimizing Eq. (1)
3. until \( \theta \) converge
4. return \( \theta \) as \( \hat{\theta} \)

---

**Algorithm 2** Online Generative Hybrid Activity Forecasting

**Require:** Trajectory simulator \( f_\pi \); Action simulator \( h_\kappa \);

1. Initialize \( f_\pi \), \( h_\kappa \) with \( \hat{\theta} \)
2. Fix all parameters except the linear layer \( \theta_0 \) at the end
3. for each new example do
   4. \([x, y, z] \leftarrow \text{slam.tracking}() \)
   5. Calculate \( H(p, q_\pi) \) with Eq. (4) (6)
   6. Sample \( z \sim \mathcal{N} \); Generate trajectories \( \hat{x} = f_\pi(z; \phi) \)
   7. Calculate \( H(q_\pi, \tilde{p}) \) with Eq. (8)
   8. Finetune \( \theta_0 \) by optimizing Eq. (9) with SGD
   9. end for

---

### 3.5. Policy Modeling

**Trajectory Modeling.** For the trajectory policy \( \pi \), we use a recurrent neural network (RNN) with gated recurrent units [3] that maps context \( \psi_t \) to \( \mu_t \) and \( S_t \). We use the matrix exponential [31] to ensure the positive definiteness of \( \sigma_t \): \( \sigma_t = \exp \left( S_t + S_t^T \right) \). The network architecture is shown in Figure 2. We provide more architectural details in the supplementary material.

**Action Modeling.** Our action policy \( \kappa \) maps context \( \chi_t \) to action probabilities \( u_t \), and is based on the idea of Temporal Segment Networks [47] with a ResNet-50 [15] backbone. The past images \( V_{t-p,t-1} \) we observe are divided into \( K \) segments and an image is selected randomly from each segment. These images are passed through a ResNet independently to get the class scores. Another fully-connected layer is built on top of the ResNet to fuse these class scores to yield segmental consensus, which serves as a useful feature in our action forecasting. In the meanwhile, the past trajectory \( x_{t-p,t-1} \) also includes useful information about what kind of actions people may perform. Thus, we add an MLP which takes the segmental consensus and the past trajectory as inputs to generate the action probabilities \( u_t \).

---

176
3.6. Prior Distribution Approximation

It is challenging to evaluate $H(q_x, p)$ without the PDF of $p$ (here, the density function of future behavior). We propose a simple approach to estimate it using the training data. For trajectory $H(q_x, p)$, we build $\bar{p}$ as a sequence of unimodal normal distributions with ground-truth trajectory $\tilde{x}$ as means, i.e., $\bar{p}(x|\phi) = N(\tilde{x}; \sigma I)$. In fact, this is identical to adding a mean squared distance penalty between the predicted trajectories and expert trajectories. For action $H(q_v, p)$, we first assume that if an action occurs at time $t$, then the same action has a higher probability happening at time steps closer to $t$. Based on this assumption, we can also view each action happening at $t$ as a unimodal normal distribution in the time dimension. If the action spans several time steps, we take the max of the distributions induced by different $t$. As a result, we obtain the approximate action prior distribution $\bar{p}(a|x, \phi)$. Note that this action prior does not actually depend on the trajectory $x$, this is partly due to the difficulty of defining a conditioned prior distribution. On the other hand, our reverse cross entropy can be seen as a regularization of trajectories and action, and the independent version can achieve this.

3.7. Online No-regret Learning

To apply the proposed framework to an online scenario where the policies are learned over time, we would like to ensure that the learning process is guaranteed to converge to the performance of the strongest model. We can evaluate the relative convergence properties of an online learning algorithm through regret analysis. To leverage known proofs of no-regret learning, one should ensure that the model and loss function being used is convex. To this end, we pretrain the network and fix parameters of nonlinear layers. We slightly adjust the trajectory reverse cross entropy as Eq. (8) and perform online gradient descent on the loss function in Eq. (9) by fine-tuning the parameters of the last linear layer. The regret is computed with respect to a model family, and the model family we consider is one of pre-trained representations that are fine-tuned to adapt to online performance. The detailed online learning parameterization is explained in supplementary material.

$$H(q_x, \bar{p})_{\text{adj}} = -E_{x_1, x_2 \sim p, x_1, T \sim q_x} \log \bar{p}(x|\phi),$$  \hspace{1cm} (8)

$$L_{\text{online}} = H(p, q_x) + H(q_v, \bar{p})_{\text{adj}} + H(p, q_v). \hspace{1cm} (9)$$

In general, the regret $R_T$ of an online algorithm is defined as: $R_T = \sum_{t=1}^{T} l_t(\xi_t; \theta_t) - \min_{\theta^*} \sum_{t=1}^{T} l_t(\xi_t; \theta^*)$, where $\xi_t$ and $l_t$ is the input and the loss at time step $t$ separately. We can prove our forward cross entropy loss is convex with respect to the parameters of the finetuned linear layer. If we further constrain the parameter’s norm $\|\theta\|_2 \leq B$ and the gradient’s norm $\|\nabla \theta\|_2 \leq L$, then the regret of our online algorithm is bounded [43] as: $R_T \leq BL\sqrt{2T}$.

Since the bound is sub-linear in $T$, the average regret $R_T/T$ approaches zero as $T$ grows, so it is a no-regret algorithm. The overall online learning procedure is shown in Algorithm 2. The detailed proof of the no-regret property is given in the supplementary material and the empirical results are shown in the experiments.

4. Experiments

We evaluate our models and baselines on the EPIC-KITCHEN [3] dataset. In this section, we first describe the dataset and related data processing steps. We then introduce the baselines that we use to compare our model with and the metrics to evaluate the performance of trajectory forecasting and action forecasting. In the experiments, we perform both batch and online experiments with the goal to validate the following hypotheses: (1) Since the trajectory-action joint model make actions conditioned on positions, the extra position information should help achieve better action forecasting performance than separately trained model. (2) The reverse cross entropy terms for trajectory and actions in our loss function should help improve sample quality. (3) The ability of evaluating the exact PDF of the trajectory and action distribution should help our model achieve lower cross entropies and higher sample quality than other generative methods that do not optimize the exact PDF such as CVAE. (4) The generative model should have the ability to generate samples with higher quality than discriminative models since it considers the multi-modal nature of future behavior and can generate multiple reasonable samples during the evaluation, while discriminative models can not. (5) We want to show from an empirical perspective that our online learning method is effective and no-regret.

4.1. Data Description

We evaluate our method on the EPIC-KITCHENS dataset [4]. First, we use ORB-SLAM [30] to extract the person’s 3D positions from the egocentric videos. For each video, we start to collect positions when the pose-graph is stable and no global bundle adjustment is performed. We also scale positions with the first 30-second results by assuming that the person’s activity range in each video is similar to alleviate the scale ambiguity caused by the initialization of ORB-SLAM. Then, we extract examples with successive 7-second interval. Those discontinuous examples (such as when tracking gets lost) are dropped out. In each 7-second example, we use the past 2 seconds as context to predict the future trajectory and actions in the next 5 seconds. We down-sample original data to 5 fps for position, 2 fps for images, and 1 fps for actions. Thus, the context we use to train the model contains 10 past positions and 4 past images. We filter actions to guarantee that each action occurs at least 50 times and drop videos which includes less than 5 examples. Finally, we use 4455 examples in total,
which come from 135 videos. The number of action classes is 122 with 39 verbs and 83 nouns. Since the annotations of the test set are not available, we randomly split the original training videos to training, validation, and test with the proportion of 0.7, 0.1, 0.2. At the same time, we ensure each action occurs in both training set and test set and the examples in different sets come from different videos.

We predict verbs and nouns separately instead of predicting the pairs of them, which is different from the setting in [4]. This is because first the combination of verbs and nouns would create too many action classes and each class would have few samples; second, there are often multiple actions taking place at the same time in the dataset, which leads to our multi-label classification formulation.

4.2. Baselines and Metrics

**Baselines** The baselines we use include two generative models and a discriminative model:

- **Direct Cross Entropy (DCE):** a generative model that uses a sequence of Gaussian to model the trajectory distribution, and a sequence of Bernoulli distributions conditioned on the trajectory to model the action distribution.
- **Conditional Variational Autoencoder (CVAE):** an auto-regressive variant VAE-based generative model. We use the Gumbel-Softmax to model the action distribution.
- **Mixed Regression and Multi-label Classification (MRMC):** a discriminative model trained by minimizing the mean squared error of trajectories and the binary cross entropy of actions.

For all baseline models, we follow the same network structure as our model to process past positions and images context. Detailed info can be found in the supplementary.

**Metrics** We use the following metrics to comprehensively evaluate our method and other baselines:

- **Forward Cross entropy:** for trajectory and action forecasting, we use their corresponding forward cross entropies $H(p, q_\pi)$ and $H(p, q_a)$ to evaluate how well the policy mimics the behaviors of the expert.
- **minMSD and meanMSD:** for trajectory forecasting, we also include two common sample-based metrics often used in generative models – minMSD and meanMSD [24, 46, 14, 36]. minMSD computes the smallest distance from $K$ samples to the ground-truth $x$: \[
\min_{k} \| \hat{x}_k - x \|^2.
\] Thus, minMSD evaluates the quality of the best sample. In contrast, meanMSD evaluates the overall quality of all $K$ samples via \[
\frac{1}{K} \sum_{k=1}^{K} \| \hat{x}_k - x \|^2.
\] The combined use of these two metrics evaluates the quality of generated trajectories comprehensively. We sample 12 trajectories for each example. For discriminative models, we directly report the regression results as minMSD and meanMSD.

- **Precision, Recall and F-1 score:** for action forecasting, since the action space is large and we need to forecast actions in 5 seconds per example, the exact matching accuracy is not be a good metric. Instead, we calculate the example-based precision and recall as [56]. One special case is that if there is no ground-truth action or predicted action happening at some time step, the denominator will be zero. If this happens, the precision and recall is 1 only if $tp = fp = fn = 0$, where $tp, fp, fn$ is the number of true positives, false positives, and false negatives, otherwise the precision and recall is 0. To consider both precision and recall, we also calculate F-1 score as $F_1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$ . As action distribution is conditioned on the forecasted trajectory, we first sample 12 trajectories, and for each trajectory we sample the action (for each action class, the action happens if its logit is greater than 0.5) and average the metrics across the trajectories. For discriminative models, we directly report the multi-label classification results.

4.3. Batch Forecasting Results

Our full model is a joint forecasting model which makes actions conditioned on the trajectory, and it is trained using the complementary loss function in Eq. (1). To test whether the joint modeling of trajectory and action distribution help improve forecasting performance, we also train a trajectory forecasting model and an action forecasting model separately. We also evaluate a variant of our method by using only the forward cross entropy for both action and trajectory. The results are summarized in Table 1.

First, we can see that our joint forecasting model (g) outperforms separately trained models (e) in action forecasting metrics (cross entropy, precision, recall, and F1-score), so our factorization – conditioning actions on the trajectory indeed helps. Hypothesis (1) is supported. Comparing (e)(g) to (f)(d), we can see the quality of both trajectory samples and actions samples are better after using the reverse cross entropy, which justifies its use in the loss function and also demonstrates the effectiveness of our designed prior data distribution. Hypothesis (2) is supported. Furthermore, our methods outperforms other generative baselines (b)(c) in terms of most metrics, especially forward cross entropy. This is due to the fact that our method has more modeling power than DCE, and can evaluate the exact PDF of trajectory and action distribution instead of optimizing the variational lowerbound like CVAE does. Our model does not outperform MRMC in the meanMSD metric and DCE in the Recall metric, but we note that: 1. The MRMC model can not conduct sampling, so it leads to lower meanMSD than all of other generative models; 2. The DCE model actually cannot generate good enough examples, which is indicated by the low precision and low F1 score, even if it has high recall; 3. All baselines make actions condi-
Table 1. Batch results on the EPIC-KITCHENS dataset. For sample-based metrics, mean ± std is reported. MRMC: Mix Regression and Multi-label Classification (discriminative model). DCE: Direct Cross Entropy (generative model). CVAE: Conditional Variational Autoencoder (generative model). For our model, S denotes separate training of trajectory policy and action policy. J denotes joint training. F denotes the model is trained with forward cross entropy only. ↓/↑ denotes a metric for which lower/higher scores are better.

<table>
<thead>
<tr>
<th>Method</th>
<th>Trajectory Forecasting</th>
<th>Action Forecasting</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$H(p, q_r)$ ($\downarrow$)</td>
<td>minMSD ($\downarrow$)</td>
</tr>
<tr>
<td>(a) MRMC</td>
<td>-</td>
<td>0.392</td>
</tr>
<tr>
<td>(b) DCE</td>
<td>-26.93</td>
<td>0.539 ± 0.010</td>
</tr>
<tr>
<td>(c) CVAE</td>
<td>-129.78</td>
<td>0.319 ± 0.008</td>
</tr>
<tr>
<td>(d) Ours (S)-F</td>
<td>-288.26</td>
<td>0.304 ± 0.017</td>
</tr>
<tr>
<td>(e) Ours (S)</td>
<td>-275.81</td>
<td>0.286 ± 0.007</td>
</tr>
<tr>
<td>(f) Ours (J)-F</td>
<td>-298.92</td>
<td>0.291 ± 0.017</td>
</tr>
<tr>
<td>(g) Ours (J)</td>
<td>-298.47</td>
<td>0.293 ± 0.004</td>
</tr>
</tbody>
</table>

**Figure 3. Forecasting results visualization.** Visualization of two examples. It shows how the forecasted trajectory influences the action distribution. In each example, the left top shows observed images, the left bottom shows action distributions corresponding to two forecasted sample trajectories, and the right shows the point cloud of the scene and the forecasted trajectories (Red/Black points: Observed/Unobserved map points).

Fig. 3 shows visualization results of two examples. For each example, we show two sampled trajectories and their corresponding action distribution. In all these two examples, the forecasted trajectory influences the action distribution on positions, so it is fair to compare Ours(J) with baselines, which shows better performance except for two aforementioned special cases. Hypothesis (3) is supported. Finally, our method also performs better than the discriminative baseline MRMC, because it fails to model the multimodal nature of the future behavior. Fig. 4 illustrate this point further. We can see that our model continuously outperforms the discriminative model in terms of recall when we force the model output actions with top K ($K$ is from 1 to 10) probabilities. The visualization example shows an environment with uncertainty. Given past information, we are actually not sure which actions (wash hand, close tap, take cloth or dry hand) will happen. Our model assigns relatively high confidence on these probable future actions but the discriminative model only focuses on two actions – wash and cloth. Thus, hypothesis (4) is also supported.
We conduct two online learning experiments to verify the effectiveness of our model to learn from streaming data. We pretrain the model on the training set and perform online learning on the test set in (i), and inversely in (ii). In both experiments, we only finetune additional linear layers during online learning. Pre-online learning and online learning results are shown in Table 2. It can be seen that in both experiments, the model obtained after online learning outperforms the original model which shows the effectiveness of our online learning algorithm. Additionally, comparing (ii) with (i), we can also see that with more data observed, the relative improvement from online learning will be more significant. We also analyze the regret of our model. We train the online models and corresponding hindsight models using Eq. (9). The average regret curve of the forward experiment is shown in Fig. 5. We can see that the average regret curve converges to zero as more examples are observed, which proves that our model is no-regret. Hypothesis (5) is also supported. The theoretical analysis of no-regret can be found in the supplementary.

### 5. Conclusion

We proposed a novel generative model to represent hybrid continuous and discrete state for first-person activity forecasting. We model discrete actions conditioned on continuous trajectories and learn a deep generative model by minimizing a symmetric cross entropy loss. Our model can generate both precise and diverse future trajectories and actions based on observed past images and positions. The results on EPIC-KITCHENS dataset shows our method outperforms related generative models and discriminative models. Our model can also be easily adapted to no-regret online learning, which creates more application possibilities in complex real-world scenarios. A possible future work is the united representation of continuous and discrete variables with the help of discrete normalizing flow models, instead of factorizing the joint distribution to make actions conditioned on trajectories.

### Acknowledgement.

This work was sponsored in part by Center for Machine Learning and Health PhD Fellowship, and JST CREST (JPMJCR14E1).
References


[28] Wei-Chiu Ma, De-An Huang, Namhoon Lee, and Kris M Kitani. Forecasting interactive dynamics of pedestrians with