Generative Hybrid Representations for Activity Forecasting with No-Regret Learning Supplementary Materials

Jiaqi Guan[1, 2], Ye Yuan[1], Kris M. Kitani[1], and Nicholas Rhinehart[1, 3]

[1]Carnegie Mellon University [2]University of Illinois Urbana Champaign [3]UC Berkeley

{jiaqig, yyuan2, kkitani}@cs.cmu.edu, nrhinehart@berkeley.edu

1. Proof of Regret Bound

According to Equation 2.5 of [6], if a loss function f parameterized by θ is convex and $\|\theta\|_2 \leq B$, then the regret of online gradient descent R_T at the T-th time step is bounded as follows:

$$R_T \leq \frac{1}{2\lambda}\|\theta_t\|_2^2 + \lambda \sum_{t=1}^{T} \|\nabla_{\theta_t}\|_2^2 \,. \tag{1}$$

This regret bound depends on the norm of the sub-gradients produced by the algorithm, and thus is not satisfactory. To derive a more concrete bound, according to Corollary 2.7 of [6], if f_t is L_t-Lipschitz with respect to $\|\cdot\|_2$, and let L be such that $\frac{1}{T}\sum_{t=1}^{T} L_t^2 \leq L^2$, then we have:

$$R_T \leq \frac{1}{2\lambda}\|\theta\|_2^2 + \lambda T L^2 \,. \tag{2}$$

In particular, if $\lambda = \frac{B}{L\sqrt{2T}}$, then

$$R_T \leq BL\sqrt{2T} \,. \tag{3}$$

It is obvious that the average regret $\frac{R_T}{T}$ approaches zero as T increases since this bound is sublinear in T. Therefore, as long as we can prove the loss function is convex with respect to θ, then we can add constraints to the variable norm $\|\theta\|_2$ and the gradient norm $\|\nabla_\theta\|$, and our algorithm will be no regret. Next, we will prove the convexity of our forward and reverse cross entropy losses.

1.1. Trajectory Forecasting Loss

First, we will consider the trajectory forecasting loss. To perform online learning for our model, we pretrain the trajectory simulator f_π and fix its parameters, and only add a learnable linear layer to transform the μ_t output by the policy π, so x_t can be rewritten as

$$x_t = \hat{\mu}_t + \sigma_t z_t = x_{t-1} + \theta_\mu \mu_t + \sigma_t z_t \,. \tag{4}$$

where θ_μ is the learnable parameter. The policy output μ_t and σ_t are fixed during online learning.

Trajectory Forward Cross Entropy For the forward cross entropy, we will not use the reparameterization trick and change-of-variable formula to analyze its convexity. Instead, it can be directly written as follows:

$$\begin{aligned} H(p, q_\pi) &= \mathbb{E}_{\tilde{x}\sim p} - \log q_\pi(\tilde{x}|\phi) \\ &= \mathbb{E}_{\tilde{x}\sim p} \sum_{t=1}^{T} - \log N(\tilde{x}_t; \hat{\mu}_t, \Sigma = \sigma_t \sigma_t^T) \\ &= \mathbb{E}_{\tilde{x}\sim p} \sum_{t=1}^{T} - \log \frac{\exp(-\frac{1}{2}(\tilde{x}_t - \hat{\mu}_t)^T \Sigma^{-1} (\tilde{x}_t - \hat{\mu}_t))}{\sqrt{(2\pi)^3 |\Sigma|}} \\ &= \mathbb{E}_{\tilde{x}\sim p} \sum_{t=1}^{T} \frac{1}{2} (\log |\Sigma| + (\tilde{x}_t - \hat{\mu}_t)^T \Sigma^{-1} (\tilde{x}_t - \hat{\mu}_t) + 3 \log 2\pi) \,. \end{aligned} \tag{5}$$

Notice that since $\tilde{x}$ is sampled from p rather than q_π, we can decompose the objective function over t, and we only need to demonstrate the convexity of the loss function at each time step. For convenience, we will omit the subscript t in the following proof.

In our setting, $\tilde{x}_t, \hat{\mu}_t \in \mathbb{R}^{3\times 1}, \sigma_t \in \mathbb{R}^{3\times 3}$. If we use a general linear layer, i.e., $\theta_\mu \in \mathbb{R}^{3\times 3}$, the gradient of the objective is:

$$\begin{aligned} \nabla_{\theta_\mu} &= -\Sigma^{-1}(x_t - \hat{\mu}_t)\mu_t^T \\ &= -\Sigma^{-1}(x_t - (x_{t-1} + \theta_\mu \mu_t))\mu_t^T \,. \end{aligned} \tag{6}$$

The Hessian matrix of the objective is

$$\mathbf{H} = \nabla^2_{\theta_\mu} = (\mu\mu^T \otimes \Sigma^{-1}) \,, \tag{7}$$

where $\otimes$ denotes the Kronecker product and $\mathbf{H} \in \mathbb{R}^{9\times 9}$. We use A to denote $\mu\mu^T$ and B to denote Σ^{-1}. With the properties of the Kronecker product and the trace operator, we can prove the positive semidefiniteness of $\mathbf{H}$ as follows: for any $x \in \mathbb{R}^{3\times 3}$, we have

$$\begin{aligned} &\text{vec}(x^T) \cdot (A \otimes B) \cdot \text{vec}(x) \\ =&\text{vec}(x^T) \cdot \text{vec}(AxB) \\ =&\text{Tr}(x^T AxB) \\ =&\text{Tr}(x^T \mu\mu^T x \Sigma^{-1}) \\ =&\text{Tr}(\mu^T x \Sigma^{-1} x^T \mu) \\ =&(x^T\mu)^T \Sigma^{-1} (x^T \mu) \\ \geq&0 \,, \end{aligned} \tag{8}$$

where we use the fact that since $\Sigma = \sigma_t \sigma_t^T$ is positive definite, its inverse Σ^{-1} is also positive definite. Because the Hessian of the objective is positive semidefinite, our algorithm's forward cross entropy loss of trajectory forecasting is convex with regard to the linear layer's parameter θ_μ.

Trajectory Reverse Cross Entropy The original reverse cross entropy of trajectory forecasting can be written as follows:

$$\begin{aligned} H(q_\pi, \tilde{p}) &= -\mathbb{E}_{x\sim q_\pi} \log \tilde{p}(x) \\ &= -\mathbb{E}_{x\sim q_\pi} \sum_{t=1}^{T} \log N(x_t; \mu = \tilde{x}_t, \Sigma = \lambda I) \,. \end{aligned} \tag{9}$$

Notice that x_t is generated from μ_t and σ_t, which are functions of past generated positions $x_{t-1-H:t-1}$. So the reverse cross entropy contains complex nonlinear operations and it is hard to guarantee its convexity. To tackle this problem, at each time step t, we sample past trajectories $x_{1:t-1}$ from the true data distribution and only sample position x_t from our policy, i.e. $x_{1:t-1} \sim p, x_t \sim q_\pi$. If we write down the reverse cross entropy under this setting, we will find it is $c_1(\mu_t + \sigma_t z_t - x_t)^T \Sigma^{-1}(\mu_t + \sigma_t z_t - x_t) + c_2$ (c_1, c_2 are constants), which only differs from the forward cross entropy $(x_t - \mu_t)^T \Sigma^{-1}(x_t - \mu_t)$ with a constant term. Thus, we can also prove the convexity of the reverse cross entropy using the convexity of the forward cross entropy.

1.2. Action Forecasting Loss

Action Forward Cross Entropy Recall that the forward cross entropy loss of action forecasting is defined as

$$\begin{aligned} H\left(p, q_{\kappa}\right) &= -\mathbb{E}_{(x,a)\sim p} \log q\left(a|x, \phi\right) \\ &= -\mathbb{E}_{(x,a)\sim p} \sum_{t,c} \log \tau \left(\sum_{i=1}^{2} \frac{u_{t,c,i}\left(\chi_t\right)}{a_{t,c,i}^{\tau}} \right)^{-2} \prod_{i=1}^{2} \left(\frac{u_{t,c,i}\left(\chi_t\right)}{a_{t,c,i}^{\tau+1}} \right), \end{aligned} \tag{10}$$

where κ is our action policy (which maps context χ to action logits u), a is the true action label, and τ is the Gumbel-Softmax temperature.

Similar to trajectory forecasting, we apply an affine transformation on u. Since a is sampled from p and there is no correlationship among actions if we apply the action-wise affine transformation, we can decompose the loss function over time and actions, and simply analyze the convexity of the loss for a single action class c at a single time step t. Thus, we drop the subscripts t, c and use u_1, u_2 to represent the action probabilities output by the policy κ. Since u_1, u_2 are generated by the softmax operation on the last layer's output v_1, v_2, we use parameters θ_1, θ_2 to transform the last layer before the softmax operation, and the new action probabilities are defined as

$$\begin{aligned} u_1 &= \frac{e^{\theta_1 v_1}}{e^{\theta_1 v_1} + e^{\theta_2 v_2}} = \frac{w_1}{w_1 + w_2}, \\ u_2 &= \frac{e^{\theta_2 v_2}}{e^{\theta_1 v_1} + e^{\theta_2 v_2}} = \frac{w_2}{w_1 + w_2}. \end{aligned} \tag{11}$$

Thus, the action forward cross entropy for a single action class c at time step t can be written as

$$\begin{aligned} H(\theta_1, \theta_2) &= -\log(c_1 u_1 + c_2 u_2)^{-2} (c_1' u_1)(c_2' u_2) \\ &= -\log \frac{w_1 w_2}{(c_1 w_1 + c_2 w_2)^2} \\ &= -\log w_1 w_2 + 2\log(c_1 w_1 + c_2 w_2), \end{aligned} \tag{12}$$

where $c_1 = a_1^{\tau}, c_2 = a_2^{\tau}, c_1' = a_1^{\tau+1}, c_2' = a_2^{\tau+1}$. Since $-\log w_1 w_2 = -(\theta_1 v_1 + \theta_2 v_2)$ is a linear function, which is clearly convex, we only need to prove the convexity of $\log(c_1 w_1 + c_2 w_2)$ (sum is an operation preserving convexity). The Hessian matrix of $\log(c_1 w_1 + c_2 w_2)$ is:

$$\mathbf{H} = \begin{bmatrix} \frac{c_1 c_2 v_1^2 A}{S^2} & \frac{-c_1 c_2 v_1 v_2 A}{S^2} \\ \frac{-c_1 c_2 v_1 v_2 A}{S^2} & \frac{c_1 c_2 v_2^2 A}{S^2} \end{bmatrix}, \tag{13}$$

where $A = e^{\theta_1 v_1 + \theta_2 v_2}, S = c_1 e^{\theta_1 v_1} + c_2 e^{\theta_2 v_2}$. We can prove its positive semidefiniteness by definition: for any $x \in \mathbb{R}^2$, we have

$$\begin{aligned} x^T \mathbf{H} x &= \frac{c_1 c_2 A}{S^2} \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T \begin{bmatrix} v_1^2 & -v_1 v_2 \\ -v_1 v_2 & v_2^2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \\ &= \frac{c_1 c_2 A}{S^2} (v_1 x_1 - v_2 x_2)^2 \geq 0. \end{aligned}$$

Thus, our algorithm's forward cross entropy loss of action forecasting is also convex.

In summary, when we apply above linear transformations on the networks, our trajectory forward cross entropy, modified trajectory reverse cross entropy and action forward cross entropy are convex with respect to the parameters of the transformations. As a result, our model can perform online learning with the sum of these losses with theoretical guarantee of no-regret.

2. Network Architecture Details

Component	Input[dimensionality]	Layer or Operation	Output[dimensionality]	Details
Trajectory Simulator				
Traj	$[K, B, P, 3]$	RNN	$[K, B, 100]$	GRU cell, tanh activation
TrajFeat	$[K, B, 100]$	FC	$[K, B, 200]$	ReLU activation
TrajFeat	$[K, B, 200]$	FC	$[K, B, 12]$	Identity activation $\rightarrow \mu \in \mathbb{R}^3, s \in \mathbb{R}^{3\times3}$
Trajectory generation: $x_t = x_{t-1} + \mu_t + \text{expm}(\text{softclip}(s_t)) \cdot z_t, \quad z_t \sim \mathcal{N}$				
Action Simulator				
Image	$[B, 4, H, W, 3]$	ResNet-50 [1]	$[B, 4, 400]$	
ImageFeat	$[B, 4, 400]$	FC	$[B, 400]$	ReLU $\rightarrow$ ImageConsensus (a)
Traj	$[K, B, P, 3]$	FC	$[K, B, 200]$	ReLU
TrajEnc	$[K, B, 200]$	FC	$[K, B, 200]$	ReLU $\rightarrow$ TrajEncoding (b)
TrajFeat, ActFeat	a, b	Tile$(a) \oplus (b)$	$[K, B, 600]$	Concatenate($\oplus$)
JointFeat	$[K, B, 600]$	FC	$[K, B, 500]$	ReLU
JointFeat	$[K, B, 500]$	FC	$[K, B, C_a \times 2]$	Identity $\rightarrow v$
Action logits $[K, B, C_a \times 2]$	Softmax	$[K, B, C_a \times 2]$	Action Probability u	
Action generation: $a_{t,c} = \text{softmax}((\log(u_{t,c}) + g_{t,c})/\tau), \quad g_{t,c} \sim \mathcal{G}$				

Table 1. **Network architecture details.** Layers are arranged from top to bottom. We use the following hyper-parameters: sample number $K = 12$, batch size $B = 10$, past trajectory context horizon $P = 10$, image size $H = W = 224$, and the number of action classes $C_a = 122$.

2.1. Trajectory Network

Recall our trajectory simulator f_π:

$$x_t = \mu_t\left(\psi_t; \theta\right) + \sigma_t\left(\psi_t; \theta\right) z_t\,.$$

We assume that people tend to be still, so we model μ_t as: $\mu_t = x_{t-1} + \bar{\mu}_t$, where $\bar{\mu}_t$ can be interpreted as a *velocity*. To ensure positive-definiteness of σ_t, we use the matrix exponential: $\sigma_t = \text{expm}(s_t + s_t^T)$. To enhance numerical stability, we soft-clip s_t before calculating σ_t with the following formula: $\text{softclip}(s, L) = \frac{s}{\log \sum \exp(\text{softmax}(1, \|s\|/L))}$ (we use $L = 5$) and also add a minimum precision identity matrix ϵI to σ_t before calculating the inverse of σ_t.

We use a GRU to encode past positions $x_{t-P:t-1} \in \mathbb{R}^{10\times3}$ with 100 hidden units and a 2-layer MLP to generate $\bar{\mu}_t \in \mathbb{R}^3$ and $\sigma_t \in \mathbb{R}^{3\times3}$ with 200 hidden units. The activation function is tanh for the GRU and ReLU for the MLP. The network architecture details can also be found in Table 1.

2.2. Action Network

The ConvNet we use in our action network is ResNet-50 [1] and the network architecture is mainly based on Temporal Segment Networks (TSN) [8]. Past observed images of 2 seconds are sampled with 2 fps and cropped to 224×224. As a result, 4 past images in total are passed through the ConvNet separately. The output of the ConvNet is 400 dimensional for each image, and then a stacked 1600 dimensional feature is passed through a fully-connected layer to generate a 400 dimensional segmental consensus. Our action network also uses the past trajectory $x \in \mathbb{R}^{P\times3}$, which are passed through an MLP with a single 200-dim hidden layer with ReLU activation to output a 200-dim encoding. Finally, a 2-layer MLP takes the segmental consensus and the past trajetory encoding as input and outputs the action class scores. For this MLP, the hidden layer has 500 units and also use ReLU activation function. The output of the action network is $a \in \mathbb{R}^{C_a\times2}$. In our setting, $P = 10, T = 5, C_a = 122$. In one of the baseline, the trajectory-independent action forecasting model (*Ours(S)*), the action network does not depend on trajectories, so the action network only takes the segmental consensus as input in that setting.

The CVAE baseline follows the same network structure as our model to encode context and decode context to generate positions and actions at each time step. The only difference is that the CVAE baseline uses the context encoding to generate the mean and standard deviation of latent variables with one fully connected layer. Following [4], another fully connected layer with sigmoid activation is applied to map latent variables to a feature of the same dimensions as the context encoding. They are combined via element-wise multiplication.

3. Other Details

3.1. Evaluation Metrics

We calculate the example-based precision and recall in the same way as [9]:

$$\text{precision} = \frac{1}{NT_a} \sum_{N,T_a} \frac{\Sigma_{C_a} tp}{\Sigma_{Ca}(tp + fp)} ,$$

$$\text{recall} = \frac{1}{NT_a} \sum_{N,T_a} \frac{\Sigma_{C_a} tp}{\Sigma_{C_a}(tp + fn)} ,$$

where N is the number of examples, and T_a is the action forecasting horizon. tp, fp, fn is the number of true positives, false positives, and false negatives respectively.

3.2. Data Augmentation

Data augmentation can generate diverse training samples and alleviate over-fitting. In our implementation, we follow the same approach used in VGG [7]: resize the original image to 256×256, randomly crop it to 224×224, and randomly flip it horizontally. Additionally, since we down-sample videos with a lower fps, we split the original sequence of images into 4 snippets with equal length, and randomly select images from each snippet during the training phase. This is similar to the approach used in TSN [8].

3.3. Data Perturbation

As mentioned in [5], $H(p, q)$ is lower-bounded by $H(p)$, but it may be unbounded below since $H(p)$ may be arbitrarily negative. Thus, we also perturb the trajectories in training data with $-\mathbb{E}_{\eta\sim\mathcal{N}(0,\eta I)}\mathbb{E}_{x\sim p} \log q(x + \eta)$ and $\eta = 0.0001$. It eliminates singularity of the metric, because the perturbation distribution has a finite entropy.

3.4. Training Details

In our prior distribution approximation, for trajectory reverse cross entropy $H(q_\pi, \tilde{p})$, we build $\tilde{p}$ as a sequence of unimodal normal distributions with ground-truth trajectory $\tilde{x}$ as means, *i.e.*, $\tilde{p}(x|\phi) = N(\cdot|\tilde{x}; \sigma I)$. We choose $\sigma = 0.01$. For action $H(q_\kappa, p)$, we also view each action happening at t as a unimodal normal distribution in the time dimension. We choose the 0.5 as the scale factor. For the model trained with both forward and reverse cross entropy losses, we use $\beta = 0.02$ for the trajectory reverse cross entropy and $\beta = 0.1$ for the action reverse cross entropy.

We use Adam [2] as the optimizer to perform gradient descent. The learning rate is set to 1e-4. The gradient is cut off from the action network to the trajectory network. When training the joint model, we use ResNet pretrained on ImageNet to initialize our ConvNet, and we also pretrain our trajectory network with trajectory forecasting. The number of epochs is 50 for training trajectory our joint model and 300 for training the separate model. The batch size is 16 and the sample number of each data is 12. We report the results on the test set using the model that performs the best on the validation set.

4. Additional Evaluation Results

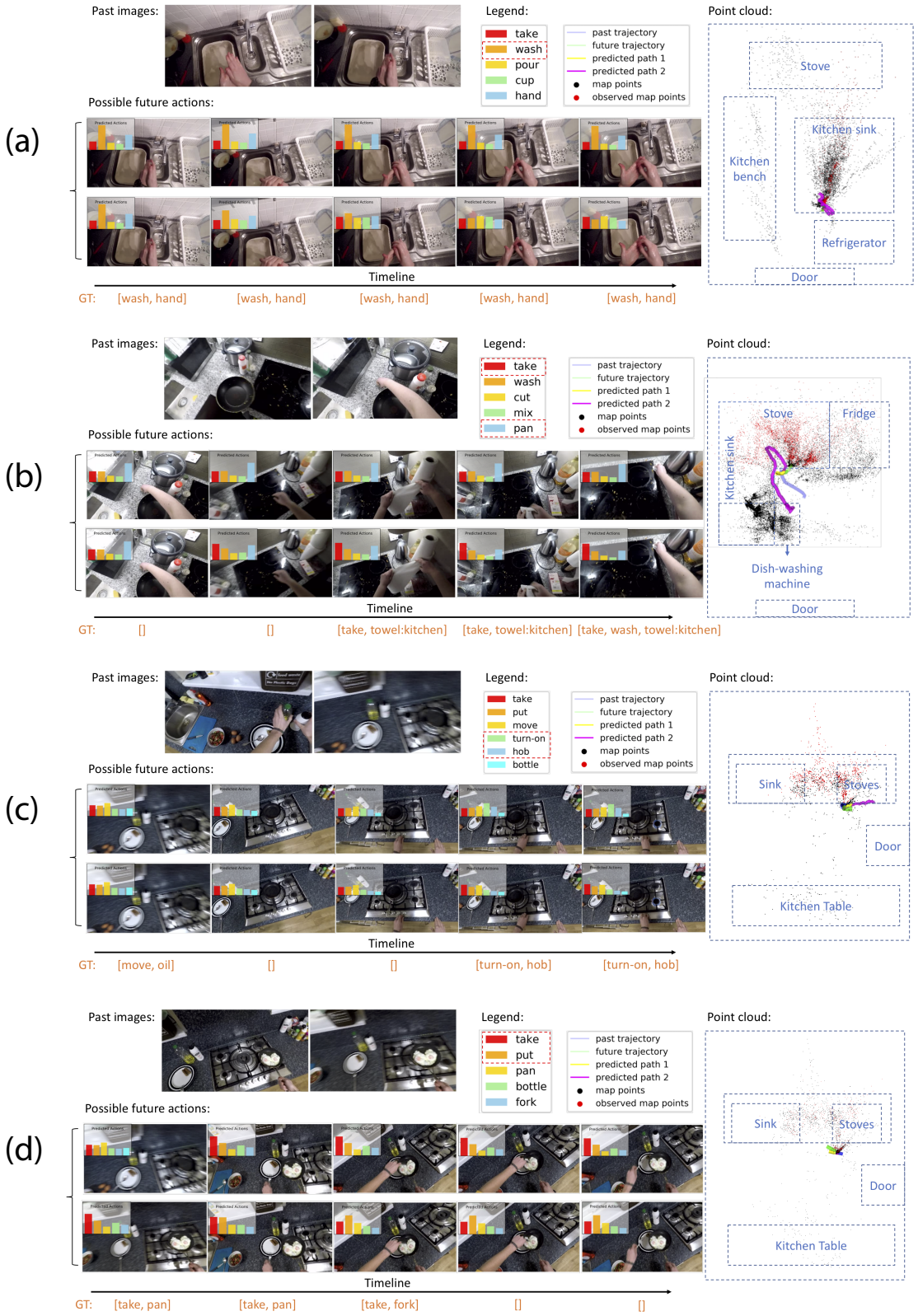

Figure 1. **Forecasting results visualization.** The figure shows the additional visualization results of four examples. It shows how the forecasted trajectory influences the action distribution. In each example, the left top shows the observed images in the past 2 seconds, the left bottom shows the action distributions corresponding to two forecasted sample trajectories, and the right shows the point cloud of the scene and the forecasted trajectories.

4.1. Diversity Analysis

Our model can generate more diverse samples than the CVAE and the discriminative model, as measured by two new metrics of diversity:

- The number of similar action sequences (*N.Act*). Two time-related (TR) action sequences count as similar if action a occurs at time t in both sequences. Two time-unrelated (TU) action sequences count as similar if action a occurs at any time in both sequences. Similar to [3], we apply temperature scaling on the Gumbel noise g with a factor of 0.3.
- The cosine similarity of generated samples (*CoSim*). We use this metric to evaluate both actions and trajectories. We report the mean cosine similarity between $\binom{12}{2}$ pairs. We also report another number in parentheses by setting a similarity threshold 0.3 to determine whether two samples are different and count the number of different samples.

	Trajectory	Action			
Method	Traj CoSim (↓)	N.Act (TR) (↑)	N.Act (TU) (↑)	Act CoSim (TR) (↓)	Act CoSim (TU) (↓)
MRMC	–	0.548	1.236	–	–
CVAE	0.232 (2.83)	0.772	1.419	0.618 (2.05)	0.634 (2.39)
Ours	**0.168 (4.13)**	**1.651**	**7.463**	**0.291 (4.38)**	**0.115 (8.83)**

Table 2. **Sample diversity evaluation results.** (↓)/(↑) denotes a metric for which lower/higher scores are better.

4.2. Visualization Results

Fig. 1 shows additional visualization results of four examples. For each example, we show two sampled trajectories and their corresponding action distribution. In all these examples, the forecasted trajectory influences the action distribution in a meaningful way. In the first example (a), the person is washing hands. The *wash* action will be more likely to happen if the person stays still, just as indicated by the first sampled trajectory and its action distribution. If the person moves a lot such as in the second sampled trajectory, the model predicts the wash action is less likely to happen. In the second example (b), the person moves less in the first forecasted trajectory than the second one. Thus, in the first trajectory, the object the person interacts with in the observed frames, i.e. *pan*, will have a high probability in the future. In the second trajectory, although the model does not predict the ground-truth object *towel:kitchen* correctly, the probability of *take* increases and the probability of *pan* decreases, which indicates the person may take something else because the person moves more in this forecasted trajectory. In the third example (c), the probability of *turn-on* and *hob* increases when the person moves and then stops, as shown in the first sampled trajectory. However, if the person keeps moving like in the second trajectory, these two actions will both have a low probability. In the fourth example (d), the large range of movement like the first sampled trajectory will lead to a higher probability of *take* than *put*, because when taking something the person has to move around to fetch the item.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 4

[2] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[3] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pages 10236–10245, 2018. 7

[4] Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B Choy, Philip HS Torr, and Manmohan Chandraker. Desire: Distant future prediction in dynamic scenes with interacting agents. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 336–345, 2017. 4

[5] Nicholas Rhinehart, Kris M Kitani, and Paul Vernaza. R2p2: A reparameterized pushforward policy for diverse, precise generative path forecasting. In *European Conference on Computer Vision*, pages 794–811. Springer, Cham, 2018. 5

[6] Shai Shalev-Shwartz et al. Online learning and online convex optimization. *Foundations and Trends® in Machine Learning*, 4(2):107–194, 2012. 1

[7] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 5

[8] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *European Conference on Computer Vision*, pages 20–36. Springer, 2016. 4, 5

[9] Min-Ling Zhang and Zhi-Hua Zhou. A review on multi-label learning algorithms. *IEEE transactions on knowledge and data engineering*, 26(8):1819–1837, 2014. 5