Supplementary Material for R2P2: A ReparameteRized Pushforward Policy for Diverse, Precise Generative Path Forecasting

Nicholas Rhinehart[1,2], Kris M. Kitani[1], Paul Vernaza[2]

[1] Carnegie Mellon University, Pittsburgh PA 15213, USA
[2] N.E.C Labs America, Cupertino, CA 95014, USA

1 Video Description

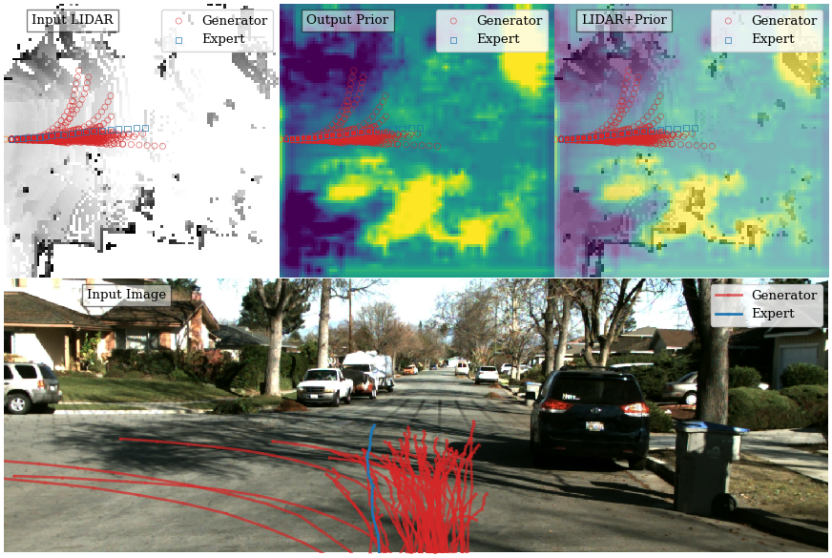


Fig. 1: Example video frame. *Top row:* Overhead views depict generated and true trajectories on each scalar map. *Top left:* The input LIDAR map is shown. *Top middle:* The prior's "cost map" is shown. *Top right:* The approximate prior and the input LIDAR map are alpha blended. *Bottom:* Our generated trajectories (red) and the true predicted trajectory (blue) are shown after projection to the image frame.

Please view the video included with the supplementary material. The video was generated by running our method on held-out test data from our newly collected CaliForecasting dataset. In Figure 1, we show an example frame from the video. The top row depicts various overhead views, along with our

method's generated trajectories, and the bottom row depicts the natural image input. Our method's generated trajectories and the true future trajectory are shown in each view.

2 Properties of the cross-entropy metric

2.1 Lower-bounding the perturbed cross-entropy

The naive cross-entropy metric $-\mathbb{E}_{x\sim p}\log q(x)$ is unbounded below when the entropy of p is unbounded below. This happens in practice, for example, when the data deterministically satisfies some linear or nonlinear constraint. To avoid this problem, we employ a perturbed version of the cross-entropy metric for both training and evaluation. Specifically, we use the metric

$$-\mathbb{E}_{\eta\sim\mu}\mathbb{E}_{x\sim p}\log q(x-\eta). \tag{1}$$

In practice, we choose the perturbation distribution μ to be Gaussian with zero mean and covariance ϵI for simplicity.

We now show that the value of the perturbed cross-entropy is lower-bounded by the entropy of the perturbation distribution, which eliminates singularity of the metric, as long as the perturbation distribution has finite entropy. We first note that Eq. (1) can be written as $H(\tilde{p}, q)$, where $\tilde{p}$ is the convolution of p with μ: $\tilde{p}(x) \triangleq \int p(y)\mu(x-y)\,\mathrm{d}y = \mathbb{E}_{y\sim p}\mu(x-y)$. Gibbs' inequality implies $H(\tilde{p}, q) \geq H(\tilde{p})$, so it is sufficient to show that $H(\tilde{p}) \geq H(\mu)$. Observe that

$$\begin{aligned} H(\tilde{p}) &= -\int \mathbb{E}_{y\sim p}\mu(x-y)\log\mathbb{E}_{y\sim p}\mu(x-y)\,\mathrm{d}x && (2)\\ &\geq -\mathbb{E}_{y\sim p}\int \mu(x-y)\log\mu(x-y)\,\mathrm{d}x && (3)\\ &= -\mathbb{E}_{y\sim p}\int \mu(z)\log\mu(z)\,\mathrm{d}z && (4)\\ &= H(\mu), && (5)\end{aligned}$$

where the inequality results from applying Jensen's inequality to the entropy function (which is concave), and we subsequently applied the change of variables $z = x - y$. Note that we have applied Jensen's inequality in the following way:

$$E_{y\sim p}H(\mu(\cdot - y)) \leq H(E_{y\sim p}\mu(\cdot - y)), \tag{6}$$

where $\mu(\cdot - y)$ denotes the distribution $\tilde{\mu}(x|y) := \mu(x-y)$.

2.2 Cross-entropy is not coordinate-invariant

Some care must be taken when reporting cross-entropy values because cross-entropy is not coordinate-invariant; this implies that the same model will achieve different cross-entropy values depending on what units the data are expressed

in, for example. Fortunately, it is fairly simple to compute how the cross-entropy changes under coordinate transformations. Suppose $p : X \to \mathbb{R}^+$ and $q : X \to \mathbb{R}^+$ are distributions on X, and $f : X \to Z$ is a differentiable, invertible coordinate transformation from X to Z. We wish to compute $H(f_*p, f_*q)$ in terms of $H(p, q)$, where $f_*\mu$ represents the pushforward measure of μ under the map f. Using the notation $\mathrm{d}f_x$ to represent the Jacobian of f evaluated at point x, direct computation shows:

$$H(f_*p, f_*q) = -\int p(f^{-1}(z))|\det \mathrm{d}f_{f^{-1}(z)}|^{-1} \log\left(q(f^{-1}(z))|\det \mathrm{d}f_{f^{-1}(z)}|^{-1}\right) \mathrm{d}z \tag{7}$$

$$= -\int p(x) \log(q(x))|\det \mathrm{d}f_x|^{-1} \mathrm{d}x \tag{8}$$

$$= H(p, q) + \mathbb{E}_{x\sim p} \log|\det \mathrm{d}f_x|, \tag{9}$$

where the second line follows from using f to change variables in the integral from z to x. For example, if $z = f(x) = cx$, and $x \in \mathbb{R}^d$, then $\mathbb{E}_{x\sim p} \log|\det \mathrm{d}f_x| = d \log|c|$. Therefore, we could make $H(f_*p, f_*q)$ arbitrarily negative by setting c to a very small positive number.

3 Explanation of Fig. 2

Fig. 2 in the main paper illustrates how the different cross-entropy metrics $H(p, q_\pi)$ and $H(q_\pi, p)$ are sensitive to different modeling errors in q_π. Here we briefly discuss how to obtain the figures shown in the annotations of that figure.

Consider, for instance, the middle-bottom figure: $H(q_\pi, p) \approx \frac{1}{2}(M'_0 - \log\epsilon)$. We can derive this by analyzing the relative supports of the *good* and *bad* versions of q_π. Suppose q_π is the good reference model and q'_π is the bad model illustrated in the middle figure. We assume q'_π is a mixture of q_π and a "rotated" version of q^R_π that rotates the support of q_π into the pictured obstacles, such that $q'_\pi(x) = 0.5q_\pi(x) + 0.5q^R_\pi(x)$. Let $A = \text{support}(q_\pi)$ and $B = \text{support}(q^R_\pi)$. Assume A and B are approximately disjoint. We then have

$$H(q'_\pi, p) = -\int q'_\pi(x) \log p(x) dx \tag{10}$$

$$= -\int_{A\cup B} (\frac{1}{2}q_\pi(x) + \frac{1}{2}q^R_\pi(x)) \log p(x) \tag{11}$$

$$\approx -\frac{1}{2}\int_A q_\pi(x) \log p(x) + -\frac{1}{2}\int_B q^R_\pi(x) \log p(x) \tag{12}$$

Assuming $\log p(x) \approx \epsilon,\ \forall x \in B$, we then have $H(q'_\pi, p) \approx \frac{1}{2}(M'_0 - \log\epsilon)$. The other figures can be derived via similar analyses.

4 Architectural details

4.1 Form of policy

Our path distribution can be thought of being *parameterized* by a continuous action-space policy, in the following way. Recall the following relationships from the main text:

$$x_t = \mu_t^\pi(\psi_t;\theta) + \sigma_t^\pi(\psi_t;\theta)z_t \tag{13}$$

$$q_\pi(x_t|\psi_t) = \mathcal{N}(x_t;\mu=\mu_t^\pi(\psi_t),\Sigma=\sigma_t^\pi(\psi_t)\sigma_t^\pi(\psi_t)^\top), \quad z_t\sim\mathcal{N}(0,I). \tag{14}$$

The output of our stochastic policy is a distribution over continuous actions: $\pi(a_{t-1}|\psi_{t-1};\theta)$. The state-state transition dynamics in general continuous-action RL problems can be written as:

$$p(x_t|\psi_{t-1}) = \int p(x_t|\psi_{t-1},a_{t-1})\pi(a_{t-1}|\psi_{t-1};\theta)\mathrm{d}a_{t-1}$$

By taking $p(x_t|\psi_{t-1},a_{t-1}) = \delta(x_t - a_{t-1})$, *i.e.*, assuming the policy can fully control the state dynamics, in that it chooses an action, and the next state is the Dirac delta function about that chosen action, we receive $p(x_t|\psi_{t-1}) = \pi(x_t|\psi_{t-1})$. This means that $q_\pi(x_t|\psi_t) = \pi(x_t|\psi_{t-1};\theta)$. Therefore, we can think of the policy as the one-step marginal $q_\pi(x_t|\psi_t) = N(x_t;\mu_t^\pi(\psi_t),\sigma_t^\pi(\psi_t)\sigma_t^\pi(\psi_t)^\top)$.

4.2 Linear

The R2P2 Linear architecture consists of two learned affine mappings $Ax + b_0$ and $Bx + b_1$, $A \in \mathbb{R}^{2\times 2H}$, $B \in \mathbb{R}^{4\times 2H}$, $b_i \in \mathbb{R}^{2H}$. Here, H is taken to be the maximum past size of 20. As discussed in the paper, we use the matrix exponential to form σ_t^π. To enhance numerical stability, we "soft-clipped" the input S of $\mathrm{expm}(S + S^\top)$ in the following elementwise transformation, which prevents σ_t^π from shrinking arbitrarily small. $\mathrm{softclip}(S,L) = \frac{S}{\mathrm{softmax}(1,\|S\|_F/L)}$. We used $L = 5$.

4.3 Field

The R2P2 Field CNN architecture parameters are shown in Tab. 1. The input H and W dimensions are downsampled to 64×64, and the output dimensions are upsampled back to 100×100. In addition to the LIDAR and semantic segmentation channels used in the input, we added an $H \times W \times 1$ channel of grid (pixel) coordinates, an $H \times W \times 1$ channel of pixel distances to the car origin in pixel coordinates, (H/2, 0). Finally, a signed distance transform feature is added to the input, which takes the road channel from the segmentation as input, and outputs the signed distance to the road at each pixel. The final input array is of shape $H \times W \times (1 + C_s + 3)$, with $H = W = 50$, 1 layer for the LIDAR map, C_s channels for the semantic segmentation, and $C_s = 18$ in the CaliForecasting model, $C_s = 12$ for the Kitti model.

Table 1: R2P2 Field Architecture. The input H and W dimensions are downsampled to 64×64, and the output dimensions are upsampled back to 100×100. s^+ stands for the softplus layer: $s^+(x) = \log(1 + \exp x)$.

Layer	1	2	3	4	5	6	7	8	9	10	11	12	13
Kernel Size	3	3	3	3	3	3	3	3	3	3	3	3	1
Dilation	1	1	1	1	2	4	8	4	2	1	1	1	1
Channels	32	32	32	32	32	32	32	32	32	32	32	32	6
Activation	s^+	s^+	s^+	s^+	s^+	s^+	s^+	s^+	s^+	s^+	tanh	tanh	Identity

4.4 RNN

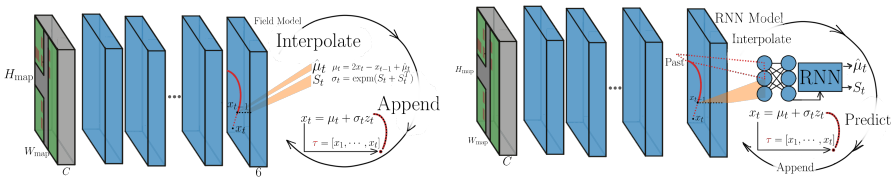


Fig. 2: RNN and CNN Policy models. The Field model produces a map of values to use for producing μ^π, σ^π through interpolation. The RNN model uses the same base as the Field model as well as information from the past trajectory to decode a featurized context representation and previous state to next μ^π, σ^π.

As discussed in the paper, the R2P2 RNN architecture builds upon the Field architecture, using the same CNN architecture settings until the last layer. Fig. 5 from the paper is reproduced here for convenience in Fig. 2. The past encoding used is produced by a GRU RNN over the past states with 150 units. The last layer from the CNN architecture is flattened and concatenated with the past encoding to form the "static" contextual input, which is passed through a 2-layer MLP with 50 hidden and 50 output units, with softplus activations. The "dynamic" input, in the sense of its dependence on t, is formed from a fixed-length zero-padded vector of flattened previous states $x_1, \ldots, x_{t-1}$. This input is passed into a GRU RNN cell to "decode" $x_1 \ldots, x_T$ to a 150 dimensional vector, which is concatenated with the "static" output to form the "joint" feature, which is passed through independent MLPs to form the μ_t^π, σ_t^π.

4.5 GAIL Discriminators

MLP Discriminator We use the MLP form as depicted in Table 2. This is the original form of the GAIL Discriminator [1]. The final activation is Softmax in the case of the original formulation, and Identity in the case of the WGAN-variant.

Table 2: GAIL Discriminators

	MLP Discriminator			CNN MLP Discriminator	
Layer	1	2	3	1	2
Units	100	100	1	128	1
Activation	tanh	tanh	Identity or softmax	tanh	Identity or softmax

CNN Discriminator In order to build a context-dependent discriminator (a function of the side information, ϕ), we used the same CNN architecture for the R2P2 Field model, except the output layer has 32 final channels. "State features" f_t are extracted from this output layer by the same bilinear interpolation process as in the Field model. These features are then passed into the MLP shown in Table 2. As before, the final activation is Softmax in the case of the original formulation, and Identity in the case of the WGAN-variant.

4.6 CVAE

We followed the architectural description of the CVAE-variant described in [2].

4.7 DCE-G

We employ the same past-encoding technique as previously described. The past encoding state is passed into a GRU RNN decoder that takes a one-hot vector that indicates the t of the rollout. The decoder produces 5-dimensional vectors, with the first two components used to parameterized μ_t^π, and the latter three components used to parameterize the upper triangle of σ_t^π.

5 R2P2 GAIL derivation

Formally, the GAIL policy seeks to optimize the following expected γ-discounted log-discriminator returns objective:

$$\max_\theta \mathbb{E}_{x\sim q_\theta} J(x) = \int dx q_\theta J(x) = \int dx q_\theta \sum_{t=1}^{T} \gamma^{t-1} \log D_\omega(x_t, a_t) \tag{15}$$

In GAIL, q_θ includes unknown model dynamics, thus $\nabla_\theta q_\theta$ cannot be computed. Optimization of Eq. 15 is done through the policy gradient shown in Eq 16. R2P2 GAIL *can* compute $\nabla_\theta q_\theta$ through its use of differentiable dynamics inside q_θ. The R2P2 GAIL gradient of Eq. 15 is shown in Eq. 17.

$$\mathbb{E}_{x\sim q_\theta}\left[J(x)\nabla_\theta \log \pi_\theta(a_t|x)\right] \tag{16}$$

$$\mathbb{E}_{z\sim q_0}\nabla_\theta J(g_\theta(z)) \tag{17}$$

Therefore, we can directly optimize the objective without relying on the indirect optimization approach of policy gradients. We provide a full derivation

of R2P2 GAIL in the supplementary material. We verify in our experiments that our approach is stabler and achieves better performance.

The GAIL training objective may be expressed as follows [3]:

$$\min_{\theta}\max_{\omega}\mathbb{E}_{\pi_\theta}\log D_\omega(s,a)+\mathbb{E}_{\pi_E}\log(1-D_\omega(s,a))-\lambda H(\pi), \tag{18}$$

where (s,a) are understood to be drawn from the marginal state-action distributions associated with either the model policy π_θ or the expert policy π_E, as indicated by subscripts of $\mathbb{E}$. The following equivalent expression makes this more explicit:

$$\min_{\theta}\max_{\omega}[\mathop{\mathbb{E}}_{\substack{s_t\sim(q_{\pi_\theta})_t\\ a_t\sim(\pi_\theta)_t\\ t\sim\mathrm{Unif}(\{1,\dots,T\})}}\log D_\omega(s_t,a_t)+\mathop{\mathbb{E}}_{\substack{s_t\sim(q_{\pi_E})_t\\ a_t\sim(\pi_E)_t\\ t\sim\mathrm{Unif}(\{1,\dots,T\})}}\log(1-D_\omega(s_t,a_t))]-\lambda H(\pi_\theta),$$

where $(q_\pi)_t$ denotes the marginal distribution of states at time t induced by rolling-out policy π, $(\pi)_t$ denotes the expected distribution of actions at time t, and Unif represents the uniform distribution. From the perspective of the outer optimization (i.e., holding ω fixed), we observe that the objective function has a form commonly treated in reinforcement learning: $\min_\theta \mathbb{E}_{x\sim q_\pi} J_\pi(x)$, where $J_\pi(x)$ is the interior expression. Using the pushforward reparameterization, the outer optimization may be written as

$$\min_{\theta}\mathop{\mathbb{E}}_{\substack{z\sim q_b\\ t\sim\mathrm{Unif}(\{1,\dots,T\})}}\log D(g_{\pi_\theta}(z)_t,\pi_\theta(z)_t)-\lambda\log H(\pi_\theta(z)_t), \tag{19}$$

where $g_{\pi_\theta}(z)_t$ denotes the roll-out of policy π_θ with random noise sequence z, evaluated at time t; and $\pi_\theta(z)_t$ denotes the policy evaluated with z at time t. Here, we regard $\pi_\theta(z)$ as a deterministic function that returns the action a_t given the random noise sequence z. Using the form of g_π suggested in the paper, we have

$$\pi_\theta(z)_t := \mu_t^\theta(x_{1:t-1},\phi)+\sigma_t^\theta(x_{1:t-1})z_t,$$

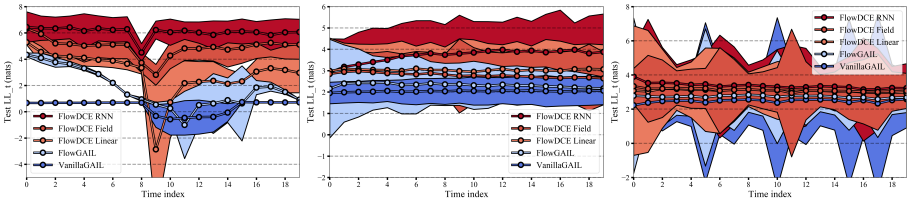
since we identify the states with the actions.

From Eq. (19), it is easy to see that the differentiability of g_{π_θ} wrt. θ trivially allows us to obtain a stochastic gradient of Eq. (19) wrt. θ by moving the derivative inside the integral.

6 Additional experiments

6.1 Time-dependent Cross-Entropy

We additionally measured the "time-dependent cross-entropy": $H(p_t, q_{\pi_t}|\hat{x}_{1:t-1}\sim p_{1:t-1}) = -\mathbb{E}_{\hat{x}\sim p}\log q_{\pi_t}(\hat{x}_t|\hat{x}_{1:t-1})$ for several approaches. Due to our model's decomposition, each time-dependent PDF $q_{\pi_t}(\cdot|\dots)$ is simply the time-specific contribution to the original joint distribution. These results are shown in Fig. 3.



(a) Cross (b) CaliForecasting (c) Kitti

Fig. 3: Test $H(p_t, q_{\pi_t})$ vs. t.

7 Derivation of Jacobian and its determinant

Given the recursive rollout equation (by recalling $\mu_t(\psi_t) = 2x_{t-1} - x_{t-2} + \hat{\mu}_t(\psi_t)$):

$$x_t = 2x_{t-1} - x_{t-2} + \mu_t(\psi_t) + \sigma_t(\psi_t) z_t,$$

then

$$J_{g_\pi}(g_\pi^{-1}(x)) = \frac{\mathrm{d}g_\pi}{\mathrm{d}g_\pi^{-1}(x)} = \begin{bmatrix} \sigma_1^\pi(\psi_1), & 0 & \dots & 0 \\ \frac{\mathrm{d}x_2}{z_1} & \sigma_2^\pi(\psi_t) & \dots & 0 \\ \vdots & & \ddots & 0 \\ \frac{\mathrm{d}x_T}{z_1} & \frac{\mathrm{d}x_T}{z_2} & \dots & \sigma_T^\pi(\psi_T) \end{bmatrix}.$$

Therefore,

$$\log \left| \det \left(\frac{\mathrm{d}g_\pi}{\mathrm{d}g_\pi^{-1}(x)} \right) \right| = \log \left| \prod_{t=1}^{T} \det(\sigma_t^\pi(\psi_t)) \right| = \sum_{t=1}^{T} \log \left| \det(\sigma_t^\pi(\psi_t)) \right|.$$

References

1. Ho, J., Ermon, S.: Generative adversarial imitation learning. In: Advances in Neural Information Processing Systems. pp. 4565–4573 (2016)
2. Lee, N., Choi, W., Vernaza, P., Choy, C.B., Torr, P.H., Chandraker, M.: Desire: Distant future prediction in dynamic scenes with interacting agents (2017)
3. Li, Y., Song, J., Ermon, S.: Infogail: Interpretable imitation learning from visual demonstrations. In: Advances in Neural Information Processing Systems. pp. 3815–3825 (2017)