Prediction by Anticipation: An Action-Conditional Prediction Method based on Interaction Learning
Supplementary Material

A. Terms in the loss function

We re-write the loss function in Eq. 4 here:

$$\mathcal{L}_t = \underbrace{\mathcal{D}(\mathbf{j}^{env}_{t+1}, \hat{\mathbf{j}}^{env}_{t+1}) + \lambda\big(1 - \text{SSIM}(\mathbf{j}^{env}_{t+1}, \hat{\mathbf{j}}^{env}_{t+1})\big)}_{\mathcal{L}^{\text{rec.}}_t} + \underbrace{\text{KL}\big(q_\phi(\mathbf{z}_t|\mathbf{o}_{1:t}, \mathbf{j}^{ego}_{t+1}, \mathbf{j}^{env}_{t+1})||p_\psi(\mathbf{z}_t|\mathbf{o}_{1:t}, \mathbf{j}^{ego}_{t+1})\big)}_{\mathcal{L}^{\text{KL}}_t}.$$

Minimizing the first and last terms is equivalent to maximizing the ELBO in Eq. 3. Considering $q_\phi(\mathbf{z}_t|\mathbf{o}_{1:t}, \mathbf{j}^{ego}_{t+1}, \mathbf{j}^{env}_{t+1}) = \mathcal{N}(\mu_\phi, \Sigma_\phi)$ and $p_\psi(\mathbf{z}_t|\mathbf{o}_{1:t}, \mathbf{j}^{ego}_{t+1}) = \mathcal{N}(\mu_\psi, \Sigma_\psi)$ as Gaussian distributions, the KL term is simply differentiable with the following terms:

$$\text{KL}\big(q_\phi(\mathbf{z}_t|\mathbf{o}_{1:t}, \mathbf{j}^{ego}_{t+1}, \mathbf{j}^{env}_{t+1}))||p_\psi(\mathbf{z}_t|\mathbf{o}_{1:t}, \mathbf{j}^{ego}_{t+1})\big) = \frac{1}{2}\Big(\text{Tr}\big(\mathbf{\Sigma}_\psi{}^{-1}\mathbf{\Sigma}_\phi\big) + \big(\mu_\psi - \mu_\phi\big)^\top \mathbf{\Sigma}_\psi{}^{-1}\big(\mu_\psi - \mu_\phi\big) + \log\big(\frac{|\mathbf{\Sigma}_\psi|}{|\mathbf{\Sigma}_\phi|}\big) - d\Big),$$

where d is the dimensionality of the stochastic latent code.

B. Ablation Study

B.1. Ablation study for the ALL measure

We provided the results of ablative study of different contributing factors on MSE and TPR/TNR in the main body of the paper. Here we provide the results for ALL, which show a similar trend as MSE and TPR/TNR in Table 4. Additionally, for the regular actions, we provide the ablation result on the code splitting idea, i.e. we provide the results for a model for which the $p_\psi(.)$ and $q_\phi(.)$ encoders have completely different sets of parameters. We refer to this model as *no split* in the tables.

Dataset →	NGSIM I-80				Argoverse			
Method	$k=1$	$k=5$	$k=10$	$k=20$	$k=1$	$k=5$	$k=10$	$k=20$
PA (no RBM)	219.4 ± 4.8	214.5 ± 5.4	207.0 ± 6.6	196.7 ± 7.3	632.7 ± 9.2	614.5 ± 4.7	589.9 ± 7.2	535.9 ± 3.9
PA (no BCDE)	232.5 ± 5.5	228.4 ± 4.9	218.9 ± 6.2	200.1 ± 6.0	657.9 ± 4.5	650.4 ± 5.6	619.8 ± 6.9	573.6 ± 7.0
PA (no ME)	230.4 ± 4.5	224.3 ± 6.9	217.6 ± 8.0	203.9 ± 5.2	651.2 ± 3.4	636.2 ± 7.2	608.3 ± 3.2	555.1 ± 6.7
PA (no split)	226.8 ± 3.7	224.5 ± 7.0	213.9 ± 7.5	199.5 ± 5.3	645.2 ± 4.7	635.9 ± 6.6	605.8 ± 4.1	559.2 ± 6.1
PA-DL (no RBM)	210.5 ± 3.6	205.8 ± 5.8	196.3 ± 6.8	180.5 ± 7.1	645.6 ± 6.2	623.3 ± 5.2	601.4 ± 6.5	557.2 ± 4.8
PA-DL (no BCDE)	218.5 ± 3.1	214.2 ± 5.1	204.6 ± 6.6	193.4 ± 6.4	668.2 ± 6.1	653.2 ± 4.4	633.1 ± 7.4	585.4 ± 7.8
PA-DL (no ME)	218.2 ± 5.0	211.5 ± 7.2	205.8 ± 4.2	195.3 ± 7.5	661.2 ± 5.5	641.8 ± 7.6	620.7 ± 5.4	577.1 ± 5.0
PA-DL (no split)	215.4 ± 3.7	209.8 ± 6.2	202.4 ± 7.4	198.2 ± 4.6	655.1 ± 5.1	637.3 ± 8.4	604.1 ± 5.1	566.6 ± 4.9

Table 5: Ablative study in terms of ALL for regular actions.

Method	ALL			
	$k=1$	$k=5$	$k=10$	$k=20$
PA (no RBM)	203.5 ± 4.5	182.9 ± 6.3	134.7 ± 3.4	99.7 ± 6.2
PA (no BCDE)	221.2 ± 6.2	211.6 ± 7.5	190.9 ± 5.0	161.2 ± 7.3
PA (no ME)	219.7 ± 6.2	205.2 ± 6.8	188.2 ± 7.2	163.8 ± 9.2
PA-DL (no RBM)	198.6 ± 5.4	172.4 ± 6.8	125.5 ± 6.2	82.3 ± 4.4
PA-DL (no BCDE)	208.9 ± 3.9	194.6 ± 5.6	178.5 ± 4.9	142.5 ± 6.8
PA-DL (no ME)	209.5 ± 4.3	191.5 ± 5.8	180.6 ± 5.0	149.8 ± 6.9

Table 6: Ablative study in terms of ALL for low-probable actions in NGSIM I-80 dataset.

B.2. Discussion about role of RBMs vs BCDE and ME:

Here we provide some insight about why the rule-based modules (RBMs) contribute in the final performance of the proposed model more than motion encoding (ME) and conditioned prior on the latent code (BCDE).

The RBMs implement the idea of factorizing the ego-action, i.e. they preserve the structure of the input frame $\mathbf{j}^{ego}_{t+1}$ in the target frame $\mathbf{j}^{env}_{t+1}$. Without them the target frame can dramatically change based on the action. The BCDE framework enables us to condition the stochastic code on the previous observation. The degradation due to the lack of BCDE is less because without such conditioning the model can still reason about the past using the deterministic code, although the unconditioned stochastic samples can become misleading in larger horizons. Also, without the ME module the model can still reason about the other agents' motion by encoding $\mathbf{o}_{1:t}$, although it will be harder for the model to encode the motion features such as speed or direction. As we can see in Fig. 6, having the difference of two consecutive frames allows effortless reasoning about the motion of other agents.

It is worth mentioning that the effect of using a conditioned prior for the latent (BCDE model) is more significant for low-probable actions, as the model with conditioned code has access to its previous predictions when estimating the parameter of the latent distribution.

B.3. SSIM term and ablation

The SSIM term helps to generate more clear images. Using the validation set the optimal weights for this term, denoted by λ^*, are $\lambda^* = 0.05$ for the NGSIM I-80 dataset and $\lambda^* = 0.1$ for the Argoverse dataset. This holds for both PA and PA-DL models.

The SSIM loss is an auxiliary term in our loss function that does not necessary appear in ELBO. Therefore, we present an ablation study on this term for each of the datasets and report the result in terms of MSE and TRP/TNR for the case we don't use SSIM in the loss function. A comparison between $\lambda = 0$, i.e. not using the SSIM term, and λ^* is presented in Table 7 .

Dataset $\rightarrow$	NGSIM I-80				Argoverse							
Method	MSE				TPR	TNR	TPR	TNR	TPR	TNR	TPR	TNR
	$k=1$	$k=5$	$k=10$	$k=20$	$k=1$		$k=5$		$k=10$		$k=20$	
PA (λ^*)	2.9 ± 0.2	4.1 ± 0.3	4.7 ± 0.3	6.2 ± 0.6	99.18	99.89	94.12	99.76	90.14	99.48	83.17	96.87
PA ($\lambda = 0$)	2.9 ± 0.3	4.0 ± 0.3	4.9 ± 0.2	6.5 ± 0.4	99.09	99.88	94.03	99.64	87.56	98.19	82.25	94.03
PA-DL (λ^*)	3.3 ± 0.2	4.6 ± 0.3	5.1 ± 0.4	6.7 ± 0.6	99.40	99.91	97.13	99.83	92.55	99.71	87.98	98.02
PA-DL ($\lambda = 0$)	3.4 ± 0.2	4.6 ± 0.2	5.2 ± 0.4	6.9 ± 0.4	99.04	99.84	96.35	99.51	90.99	99.47	85.15	96.13

Table 7: Effect of the SSIM term in terms of MSE for NGSIM I-80 and TPR/TNR for Argoverse.

As we can see in both cases, adding the the SSIM term is effective, especially for larger values of k. This is due to the fact that the error of the prediction is accumulative.

C. Implementation details of the prediction module

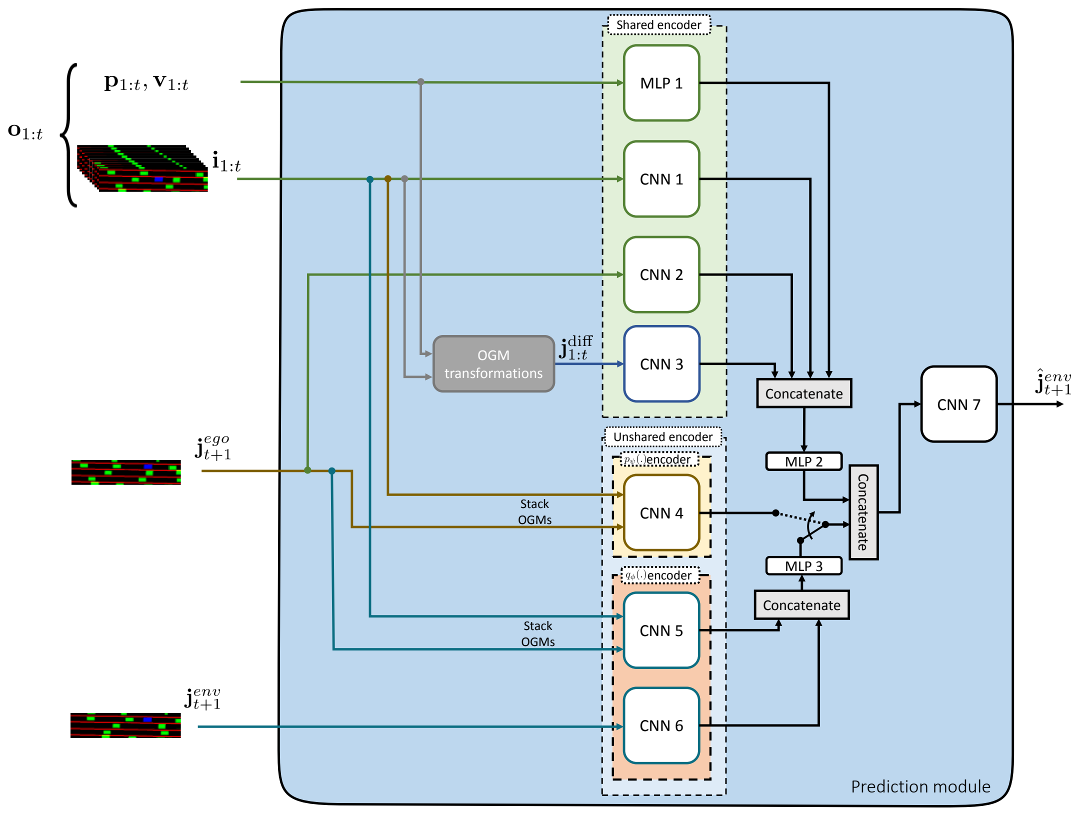

Figure 8: Detailed structure of the prediction module.

Figure 8 shows a detailed implementation of the prediction module. The structure of the networks based on this figure are explained below. Convolutional neural networks, denoted by CNN, have convolutional layers as their core but can also include one or two fully-connected layers. MLP networks only contain fully-connected layers. Also, we used ADAM optimizer for training our models with learning rate scheduler starting from lr = 0.0001.

C.1. NGSIM I-80 dataset

For the NGSIM I-80 dataset, we use the actions extracted by the authors in [12]. As stated in the experiment section, the images are $117 \times 24 \times 3$. For the IOTs we zero pad the images before feeding them to the modules with the padding size of 20 in each dimension and remove the padding from the output image after processing.

Network	Detail of structure	
MLP 1	2 layers	1. 256 units- leaky ReLU activation 2. 25 units- ReLU activation
CNN 1	4 layers	1. 64 kernels of size 4×4, stride size = 2, leaky ReLU activation 2. 128 kernels of size 4×4, stride size = 2, leaky ReLU activation 3. 256 kernels of size 4×4, stride size = 2, ReLU activation. Followed by flattening 4. Fully-connected layer with 768 units, ReLU activation
CNN 2	4 layers	1. 4 kernels of size 4×4, stride size = 2, leaky ReLU activation 2. 8 kernels of size 4×4, stride size = 2, leaky ReLU activation 3. 256 kernels of size 4×4, stride size = 2, ReLU activation. Followed by flattening 4. Fully-connected layer with 768 units, ReLU activation
CNN 3	4 layers	1. 4 kernels of size 4×4, stride size = 2, leaky ReLU activation 2. 8 kernels of size 4×4, stride size = 2, leaky ReLU activation 3. 32 kernels of size 4×4, stride size = 2, ReLU activation. Followed by flattening 4. Fully-connected layer with 32 units with linear activation as the mean, $\mu(\mathbf{j}^{\text{diff}}_{1:t})$.
CNN 4	5 layers	1. 4 kernels of size 4×4, stride size = 2, leaky ReLU activation 2. 8 kernels of size 4×4, stride size = 2, leaky ReLU activation 3. 16 kernels of size 4×4, stride size = 2, ReLU activation. Followed by flattening 4. Fully-connected layer with 768 units, linear activation 5. Two branches of fully-connected layers with 32 units each, as the mean and variance , μ_ψ and Σ_ψ, with linear activation
CNN 5	4 layers	1. 4 kernels of size 4×4, stride size = 2, leaky ReLU activation 2. 8 kernels of size 4×4, stride size = 2, leaky ReLU activation 3. 16 kernels of size 4×4, stride size = 2, ReLU activation. Followed by flattening 4. Fully-connected layer with 768 units, linear activation
CNN 6	4 layers	1. 4 kernels of size 4×4, stride size = 2, leaky ReLU activation 2. 8 kernels of size 4×4, stride size = 2, leaky ReLU activation 3. 16 kernels of size 4×4, stride size = 2, ReLU activation. Followed by flattening 4. Fully-connected layer with 768 units, linear activation
MLP 2	1 layer	1 .256 units, leaky ReLU activation
MLP 3	1 layer	1. Two branches of fully-connected layers with 32 units each, as the mean and variance , μ_ϕ and Σ_ϕ, with linear activation
CNN 7	5 layers	1. Fully-connected layer with 768 units, leaky ReLU activation 2. Fully-connected layer with 6144 units, leaky ReLU activation. Followed by reshaping to a $(12, 2, 256)$ tensor 3. Deconv. layer with 128 kernels of size 5×3, stride size = 2, leaky ReLU activation 4. Deconv layer with 64 kernels of size 6×4, stride size = 2, leaky ReLU activation 5. Deconv layer with 3 kernels of size 3×2, stride size = 2, sigmoid activation

Table 8: Detail of the prediction module for the NGSIM I-80 dataset. The coefficient for the leaky ReLU activation functions is 0.2.

For difference learning model, we use the exact same structure of the networks. However, since the output of the module is the difference between two consecutive frames, it can get any values between -1 and 1. Therefore, we use *tanh* activation for the last layer of network CNN 7.

C.2. Argoverse dataset

The images in this dataset are $256 \times 256 \times 2$. They have two channels. One channel for OGM and the other one for the ego-vehicle. The ego-vehicle is more towards the left side of the image. Therefore we first zero pad the left side of the image so that the ego-vehicle is centered and then zero pad the whole image to apply the functions in the IOTs. To extract the actions, we apply inverse of the functions in Fig. 2 to compute the actions using the xy coordinate of the car at each time. Table 9 shows the details of the difference learning model, i.e. the model with the best performance for this dataset.

Network	Detail of structure	
MLP 1	2 layers	1. 85 units- leaky ReLU activation 2. 25 units- ReLU activation
CNN 1	6 layers	1. 16 kernels of size 4×4, stride size = 2, leaky ReLU activation 2. 32 kernels of size 4×4, stride size = 2, leaky ReLU activation 3. 64 kernels of size 4×4, stride size = 2, leaky ReLU activation 4. 128 kernels of size 4×4, stride size = 2, leaky ReLU activation 5. 256 kernels of size 4×4, stride size = 2, ReLU activation. Followed by flattening 6. Fully-connected layer with 128 units, ReLU activation
CNN 2	6 layers	1. 2 kernels of size 4×4, stride size = 2, leaky ReLU activation 2. 4 kernels of size 4×4, stride size = 2, leaky ReLU activation 3. 8 kernels of size 4×4, stride size = 2, leaky ReLU activation 4. 16 kernels of size 4×4, stride size = 2, leaky ReLU activation 5. 256 kernels of size 4×4, stride size = 2, ReLU activation. Followed by flattening 6. Fully-connected layer with 128 units, ReLU activation
CNN 3	6 layers	1. 2 kernels of size 4×4, stride size = 2, leaky ReLU activation 2. 4 kernels of size 4×4, stride size = 2, leaky ReLU activation 3. 8 kernels of size 4×4, stride size = 2, leaky ReLU activation 4. 16 kernels of size 4×4, stride size = 2, leaky ReLU activation 5. 32 kernels of size 4×4, stride size = 2, ReLU activation. Followed by flattening 6. Fully-connected layer with 32 units with linear activation as the mean , $\mu(\mathbf{j}_{1:t}^{\text{diff}})$.
CNN 4	7 layers	1. 2 kernels of size 4×4, stride size = 2, leaky ReLU activation 2. 4 kernels of size 4×4, stride size = 2, leaky ReLU activation 3. 8 kernels of size 4×4, stride size = 2, leaky ReLU activation 4. 16 kernels of size 4×4, stride size = 2, leaky ReLU activation 5. 256 kernels of size 4×4, stride size = 2, ReLU activation. Followed by flattening 6. Fully-connected layer with 128 units, linear activation 7. Two branches of fully-connected layers with 32 units each, as the mean and variance , μ_ψ and Σ_ψ, with linear activation
CNN 5	6 layers	1. 2 kernels of size 4×4, stride size = 2, leaky ReLU activation 2. 4 kernels of size 4×4, stride size = 2, leaky ReLU activation 3. 8 kernels of size 4×4, stride size = 2, leaky ReLU activation 4. 16 kernels of size 4×4, stride size = 2, leaky ReLU activation 5. 256 kernels of size 4×4, stride size = 2, ReLU activation. Followed by flattening 6. Fully-connected layer with 128 units, linear activation
CNN 6	6 layers	1. 2 kernels of size 4×4, stride size = 2, leaky ReLU activation 2. 4 kernels of size 4×4, stride size = 2, leaky ReLU activation 3. 8 kernels of size 4×4, stride size = 2, leaky ReLU activation 4. 16 kernels of size 4×4, stride size = 2, leaky ReLU activation 5. 256 kernels of size 4×4, stride size = 2, ReLU activation. Followed by flattening 6. Fully-connected layer with 128 units, linear activation
MLP 2	1 layer	1 .256 units, leaky ReLU activation
MLP 3	1 layer	1. Two branches of fully-connected layers with 32 units each, as the mean and variance , μ_ϕ and Σ_ϕ, with linear activation
CNN 7	7 layers	1. Fully-connected layer with 128 units, leaky ReLU activation 2. Fully-connected layer with 9216 units, leaky ReLU activation. Followed by reshaping to a $(6, 6, 256)$ 3. Deconv. layer with 128 kernels of size 4×4, stride size = 2, leaky ReLU activation. 4. Deconv layer with 64 kernels of size 4×4, stride size = 2, leaky ReLU activation 5. Deconv. layer with 32 kernels of size 4×4, stride size = 2, leaky ReLU activation. 6. Deconv layer with 16 kernels of size 4×4, stride size = 2, leaky ReLU activation 7. Deconv layer with 2 kernels of size 3×2, stride size = 2, tanh activation

Table 9: Detail of the prediction module for the Argoverse dataset as a part of difference learning module. The coefficient for the leaky ReLU activation functions is 0.2.