NATRA: Noise-Agnostic Framework for Trajectory Prediction with Noisy Observations

Supplementary Material

6. Appendix

6.1. More analysis of NATRA

Performance under low/no noise settings. We evaluate NATRA under low or no noise by setting the Gaussian noise σ to 0.05 and 0. The results presented in Table 7 indicate that after integrating NATRA into EqMotion, the performance is still superior to baselines when at a low noise level ($\sigma = 0.05$). Additionally, NATRA+EqMotion performs comparably to EqMotion when $\sigma = 0$. This demonstrates NATRA does not degrade the performance when noise is not introduced.

Table 7. Comparison of different methods under different noise setting on the SDD dataset. The evaluation metrics are ADE and FDE (Unit: pixels). The best results are highlighted in **bold**.

Noise	Method	SDD	
		ADE	FDE
$\sigma = 0.05$	EqMotion	8.48	13.49
	Wavelet+EqMotion	8.39	13.37
	EMA+EqMotion	8.42	13.36
	NATRA+EqMotion	**8.32**	**13.28**

Noise	Method	SDD	
		ADE	FDE
$\sigma = 0$	EqMotion	**8.08**	13.12
	Wavelet+EqMotion	8.16	13.42
	EMA+EqMotion	8.22	13.57
	NATRA+EqMotion	8.11	**13.08**

Noise	Method	ETH/UCY	
		ADE	FDE
$\sigma = 0$	GraphTern	**0.24**	0.38
	Wavelet+GraphTern	0.26	0.40
	EMA+GraphTern	0.25	0.39
	NATRA+GraphTern	**0.24**	**0.37**

Performance on diffusion-based backbones. In addition to GraphTern and EqMotion, we integrate NATRA into MID, a diffusion-based model for trajectory prediction. Specifically, we first use TDM to denoise the noisy observations X_{obs}, obtaining $\hat{X}_{obs}$. Then, using both the denoised and original observations, we sample normal noise from a standard Gaussian distribution to generate $\hat{Y}_{fut}$ and $\tilde{Y}_{fut}$, respectively. To optimize the model, We apply $\mathcal{L}_{pred}$ and $\mathcal{L}_{rank}$ alongside the MID loss . The results shown in Table 8 show that NATRA still outperforms the baselines, which further underscores its adaptability.

Table 8. Comparison with baselines using MID backbone. The evaluation metrics are ADE and FDE (Unit: pixels). The best results are highlighted in **bold**.

Noise	Method	SDD	
		ADE	FDE
$\sigma = 0.4$	MID	12.86	18.35
	Wavelet+MID	12.26	17.88
	EMA+MID	12.45	18.01
	NATRA+MID	**11.97**	**17.41**

Comparison with frozen predictor. We conduct an experiment where we freeze the predictor and only train the denoiser. We first load the predictor trained on clean observations, freeze its parameters, and then integrate NATRA, training only the denoiser. The results, shown in Table 9, reveal a performance decrease when the predictor's parameters are frozen. This indicates the necessity of jointly learning the denoiser and predictor.

Table 9. Comparison with NoiseTraj where the predictor is freezed. The best results are highlighted in **bold**

Noise	Method	SDD	
		ADE	FDE
$\sigma = 0.4$	EqMotion	13.46	19.60
	NATRA+EqMotion (freeze)	12.19	17.95
	NATRA+EqMotion	**11.92**	**17.65**

Comparison with Learning-based baseline. To our knowledge, our work is the first to address trajectory prediction with noisy observations, with no existing learning-based baselines for this problem. We use Noise2Void [1], a learning-based denoiser originally for image denoising, as another baseline. We first denoise the observed trajectories, and then perform future trajectory prediction based on the observations. The results in Table 10 of the attached PDF show that NATRA outperforms Noise2Void, demonstrating the effectiveness of our method.

Hyper-parameter Analysis. We conducted hyperparameter analysis for number of masked locations. As shown in the Table 11, masking 2 trajectory locations achieves the best performance. This is because too few masked locations restrict the model's reconstruction ability, while too many make accurate recovery challenging.

Table 10. Comparison with baselines on SDD dataset. The evaluation metrics are ADE and FDE (Unit: pixels). The best results are highlighted in **bold**.

Noise	Method	SDD	
		ADE	FDE
$\sigma = 0.4$	EqMotion	13.46	19.60
	Wavelet+EqMotion	12.38	18.25
	EMA+EqMotion	12.79	18.64
	Noise2Void+EqMotion	12.46	18.52
	NATRA+EqMotion	**11.92**	**17.65**

Table 11. Analysis of number of masked locations on SDD dataset

Method	Noise	Mask Num	SDD	
			ADE	FDE
NATRA +GraphTern	$\sigma = 0.4$	1	12.77	18.01
		2	**12.35**	**17.28**
		3	13.11	18.62
		4	13.26	18.88

We also analyse trade-off parameters α, β, γ and δ. The result listed in Table 12. We conducted analysis on SDD dataset using a grid search over the set $\{0.001, 0.01, 0.1, 1.0\}$. We utilize GraphTern as the backbone. The results are listed in the table below, and we select the best set of hyperparameters as the final configuration for the model.

Table 12. Hyper-parameter analysis of trade-off α, β, γ and δ on SDD dataset.

Hyper-	0.001		0.01		0.1		1.0	
paramter	ADE	FDE	ADE	FDE	ADE	FDE	ADE	FDE
$\alpha(\mathcal{L}_{MI})$	13.31	19.04	**12.35**	**17.28**	13.86	19.74	14.98	21.06
$\gamma(\mathcal{L}_{MI})$	13.18	18.69	**12.35**	**17.28**	12.47	17.66	12.48	17.62
$\beta(\mathcal{L}_{rank})$	12.67	17.74	**12.35**	**17.28**	12.92	18.25	13.06	18.45
$\delta(\mathcal{L}_{rec})$	12.99	18.62	12.75	18.11	12.39	17.36	**12.35**	**17.28**

Analysis of noise at different timestep. We conducted experiments to evaluate the NATRA's ability to handle noise at different timestep. We divide 8-frame observations into early (first 4 frames), and late (last 4 frames) periods. We add Gaussian noise $\mathcal{N}(0, 0.4)$ to early and late period respectively to construct SDD(early) and SDD(late) dataset for evaluating NATRA. The experimental results listed in Table 14 show NATRA can still surpass baselines under noise at various time steps.

6.2. Training and Inference Time Analysis

We conduct a comparison of computational costs in terms of training time and inference latency. The results, obtained using Eqmotion as the backbone and SDD as the dataset, are listed in the Table **??**. Both $\mathcal{L}_{MI}$ and $\mathcal{L}_{rec}$ introduce minimal overhead during training. The ranking loss ($\mathcal{L}_{rank}$) requires an additional forward pass through the backbone, resulting in increased training time. However, the increase in inference time is negligible. The 10ms latency is introduced by the trajectory denoising model, while all the losses are only used during training and do not incur any additional cost at inference.

Table 13. Comparison of different methods when noise is different between training and testing on the SDD dataset. The evaluation metrics are ADE and FDE (Unit: pixels). The best results are highlighted in **bold**.

(a) Training with $\sigma = 0.4$, and Testing with $\sigma = 0.2$.

Noise	Method	SDD	
		ADE	FDE
Train: $\sigma = 0.4$ Test: $\sigma = 0.2$	EqMotion	11.47	16.82
	Wavelet+EqMotion	10.86	16.07
	EMA+EqMotion	10.99	16.24
	NATRA+EqMotion	**10.72**	**15.89**

(b) Training with $\sigma = 0.4$, and Testing with $\lambda = 0.4$.

Noise	Method	SDD	
		ADE	FDE
Train: $\sigma = 0.4$ Test: $\lambda = 0.4$	EqMotion	15.03	19.35
	Wavelet+EqMotion	14.40	18.56
	EMA+EqMotion	14.57	18.98
	NATRA+EqMotion	**14.26**	**18.32**

Table 14. Analysis of noise at different timestep on SDD(early) and SDD(late) dataset.

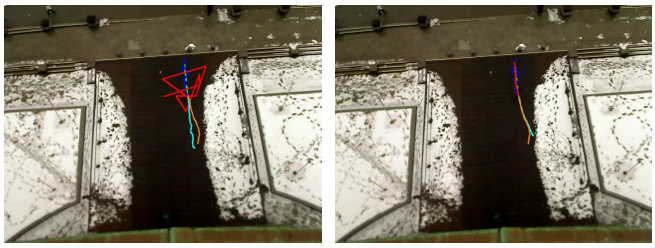
Method	SDD(early)		SDD(late)	
	ADE	FDE	ADE	FDE
GraphTern	10.12	15.36	12.14	17.68
EMA+GraphTern	9.84	15.05	11.37	16.21
Wavelet+GraphTern	9.86	15.12	11.14	16.02
NATRA+GraphTern	**9.46**	**14.48**	**10.77**	**15.65**

Table 15. Analysis of training and inference time on SDD dataset. The backbone is EqMotion

Method(on Eqmotion)	**Train(s)**	**Infer(ms)**
No denoising	565	73
Wavelet	661	97
Wavelet(Preprocess)	563	73
EMA	632	77
EMA(Preprocess)	571	73
NARTA($\mathcal{L}_{MI}$)	604	83
NARTA($\mathcal{L}_{MI}, \mathcal{L}_{rec}$)	619	83
NARTA($\mathcal{L}_{MI}, \mathcal{L}_{rec}, \mathcal{L}_{rank}$)	1119	83

6.3. Generalizability of NATRA

To verify the generalizability of our method, we conduct additional experiments where the noise in the training and



(a) Results of applying $\mathcal{L}_{MI}$ (b) Results of applying $\mathcal{L}_{MI} + \mathcal{L}_{rec}$

Figure 4. Visualization of trajectories on ETH dataset by employing (a) $\mathcal{L}_{MI}$ and (b) $\mathcal{L}_{MI} + \mathcal{L}_{rec}$. The clean, noisy, and denoised observations are shown in green, blue, and red, respectively. The ground-truth and predicted future trajectories are shown in orange and cyan, respectively.

validation/testing set is different. Specifically, we train the model using trajectories with Gaussian noise ($\sigma = 0.4$) and then test it with Gaussian noise ($\sigma = 0.2$) and Poisson noise ($\lambda = 0.4$). The results, as listed in Table 13, show that our NATRA can achieve denoising effectively, and still outperforms the baselines. This also indicates that our method possesses generalization ability when noise is different in the training and testing/validation set.

6.4. Sensitivity to the Amount of Clean GTs.

We conduct experiments to evaluate the sensitivity of NATRA to the amount of clean GT data. Specifically, we train the model on 0%, 25%, 50%, 75%, and 100% of the ground-truth trajectories on the SDD dataset. The remaining trajectories are added Gaussian noise N (0, 0.4), consistent with the noise used in the observed trajectories. The results listed in the Table 16. Results show NATRA performs comparable to baselines when using 0% and 25% amount.

6.5. Visualization of Mutual Information-Based Denoising Mechanism

To further demonstrate the efficacy of the proposed Mutual Information-Based Denoising Mechanism, we visualize the denoised trajectory and future predicted trajectory on the ETH dataset. As shown in Figure 4(a), optimizing solely for mutual information leads to the destruction of structural information. However, as depicted in the Figure 4(b), when we incorporate the reconstruction loss $\mathcal{L}_{rec}$, the structure of the trajectory is preserved, and more accurate future trajectory predictions based on these well-structured observations. This underscores the effectiveness of our proposed method.

6.6. Training Procedure of NATRA

We provide the training procedure of NATRA in the Algorithm 1.

6.7. Proof of Theorem 3.1

Theorem 6.1 (Theorem 3.1 restated)**.** *Given two random variables x and y, the mutual information $I(x;y)$ has the following upper bound*

$$I(x;y) \leq \mathbb{E}_{p(x,y)}[\log p(y|x)] - \mathbb{E}_{p(x)}\mathbb{E}_{p(y)}[\log p(y|x)] \tag{19}$$

Proof. The definition of mutual information between variables x and y is

$$\begin{aligned} I(x;y) &= \mathbb{E}_{p(x,y)}\left[\log \frac{p(x,y)}{p(x)p(y)}\right] = \mathbb{E}_{p(x,y)}\left[\log \frac{p(y|x)}{p(y)}\right] \\ &= \mathbb{E}_{p(x,y)}[\log p(y|x)] - \mathbb{E}_{p(x,y)}[\log p(y)] \\ &= \mathbb{E}_{p(x,y)}[\log p(y|x)] - \mathbb{E}_{p(y)}[\log p(y)] \end{aligned} \tag{20}$$

By the definition of the marginal distribution, we have:

$$p(y) = \int p(y|x)p(x)dx = \mathbb{E}_{p(x)}[p(y|x)]. \tag{21}$$

By substituting Equation (21) to , we have:

$$\begin{aligned} I(x;y) &= \mathbb{E}_{p(x,y)}[\log p(y|x)] - \mathbb{E}_{p(y)}[\log p(y)] \\ &= \mathbb{E}_{p(x,y)}[\log p(y|x)] - \mathbb{E}_{p(y)}[\log \mathbb{E}_{p(x)}[p(y|x)]] \end{aligned} \tag{22}$$

Note that the $\log(\cdot)$ is a concave function, by Jensen's Inequality, we have

$$\begin{aligned} -\mathbb{E}_{p(y)}[\log \mathbb{E}_{p(x)}[p(y|x)]] &\leq -\mathbb{E}_{p(y)}\mathbb{E}_{p(x)}[\log p(y|x)] \\ &= \mathbb{E}_{p(x)}\mathbb{E}_{p(y)}[\log p(y|x)] \end{aligned} \tag{23}$$

By applying this inequality to Equation (22), we obtain:

$$\begin{aligned} I(x;y) &= \mathbb{E}_{p(x,y)}[\log p(y|x)] - \mathbb{E}_{p(y)}[p(y)] \\ &= \mathbb{E}_{p(x,y)}[\log p(y|x)] - \mathbb{E}_{p(y)}[\log \mathbb{E}_{p(x)}[p(y|x)]] \\ &\leq \mathbb{E}_{p(x,y)}[\log p(y|x)] - \mathbb{E}_{p(x)}\mathbb{E}_{p(y)}[\log p(y|x)] \end{aligned} \tag{24}$$

□

Table 16. Sensitivity to the amount of clean GT data.

Method(on Eqmotion)	0%	25%	50%	75%	100%
No Denoising	14.96/20.88	14.38/20.45	14.01/20.03	13.69/19.37	13.46/19.60
Wavelet	**14.76/20.31**	**13.94/19.62**	13.34/18.97	12.85/18.52	12.38/18.25
EMA	14.87/20.65	14.01/19.96	13.45/19.34	13.03/18.84	12.79/18.64
NARTA	14.92/20.69	14.11/**19.62**	**13.29/18.7**2	**12.53/18.15**	**11.92/17.65**

Algorithm 1: Training Procedure of NATRA

Input: Noisy observations X_{obs}, ground-truth future trajectories Y_{fut}. Four trade-off hyper-parameters: α, β, δ and γ.
Output: Network parameters: $\Phi_{\textbf{TDM}}$, $\Phi_{\textbf{TPB}}$, ψ, and ϕ.
Initialize: Randomly initialize $\Phi_{\textbf{TDM}}$, $\Phi_{\textbf{TPB}}$, ψ, and ϕ.
while *Model not converges* **do**
Random mask the noisy observations using the mask vector: $X_{obs}^{mask} = X_{obs} \odot \mathcal{M}_{obs}$
Obtain the trajectories $\hat{X}_{obs}^{mask} = \Phi_{\textbf{TDM}}(X_{obs}^{mask})$
Calculate reconstruction loss $\mathcal{L}_{rec} = ||\hat{X}_{obs}^{mask} \odot (1 - \mathcal{M}_{obs}) - X_{obs} \odot (1 - \mathcal{M}_{obs})||_2$
Input noisy observations to $\Phi_{\textbf{TDM}}$ for denoising: $\hat{X}_{obs} = \Phi_{\textbf{TDM}}(X_{obs})$
Employ Mutual Information-based mechanism for further denoising:
$\mathcal{L}_{MI} = \alpha \mathbb{E}_{p(X_{obs}, \hat{X}_{obs})}[\log q_\phi(\hat{X}_{obs}|X_{obs})] - \mathbb{E}_{p(X_{obs})}\mathbb{E}_{p(\hat{X}_{obs})}[\log q_\phi(\hat{X}_{obs}|X_{obs})]$
$- \sup_{\psi} \mathbb{E}_{p(\hat{X}_{obs}, Y_{fut})}[T_\psi] + \log \mathbb{E}_{p(\hat{X}_{obs})p(Y_{fut})}[e^{T_\psi}]$
Obtain the future predictions based on denoised observations: $\{\hat{Y}_{fut}^k\}_{k=1}^K = \Phi_{\textbf{TPB}}(\hat{X}_{obs})$
Obtain the future predictions based on noisy observation: $\{\tilde{Y}_{fut}^k\}_{k=1}^K = \Phi_{\textbf{TPB}}(X_{obs})$
Calculate $d_{denoise}$ and d_{noise}:
$d_{denoise} = \min_{1 \le k \le K} ||\hat{Y}_{fut}^k - Y_{fut}||_2, \quad d_{noise} = \min_{1 \le k \le K} ||\tilde{Y}_{fut}^k - Y_{fut}||_2$
Calculate $\mathcal{L}_{pred}$ and $\mathcal{L}_{rank}$ as
$\mathcal{L}_{pred} = ||\hat{Y}_{fut}^{best} - Y_{fut}||_2 + ||\tilde{Y}_{fut}^{best} - Y_{fut}||, \quad \mathcal{L}_{rank} = \max(0, d_{denoise} - d_{noise} + \Delta)$
Optimizing $\mathcal{L} = \mathcal{L}_{pred} + \beta \mathcal{L}_{rank} + \delta \mathcal{L}_{rec} + \gamma \mathcal{L}_{MI}$ by gradient descent to update the $\Phi_{\textbf{TDM}}$ and $\Phi_{\textbf{TPB}}$.
end

6.8. Proof of Theorem 3.2

Theorem 6.2 (Therorem 3.2 restated)**.** *Given two probability distributions $\mathbb{P}$, $\mathbb{Q}$. The Kullback Liebler Divergence admits the following dual representation:*

$$D_{KL}(\mathbb{P}||\mathbb{Q}) = \sup_{T:\Omega \to \mathbb{R}} \mathbb{E}_{\mathbb{P}}[T] - \log \mathbb{E}_{\mathbb{Q}}[e^T], \tag{25}$$

Proof. The proof comprises two steps. Firstly, we prove the existence of the supremum in the dual representation. Subsequently, we demonstrate that this representation serves as the lower bound of the Kullback-Liebler Divergence.

Lemma 1. *There exist a function $T^* : \Omega \to \mathbb{R}$, such that:*

$$D_{KL}(\mathbb{P}||\mathbb{Q}) = \mathbb{E}_{\mathbb{P}}[T^*] - \log \mathbb{E}_{\mathbb{Q}}[e^{T^*}] \tag{26}$$

Proof. We choose a function $T^* = \log \frac{\mathbb{P}}{\mathbb{Q}}$, then we have:

$$\mathbb{E}_{\mathbb{P}}(T^*) - \log \mathbb{E}_{\mathbb{Q}}[e^{T^*}] = \mathbb{E}_{\mathbb{P}}\left[\log \frac{\mathbb{P}(\omega)}{\mathbb{Q}(\omega)}\right] - \log \mathbb{E}_{\mathbb{Q}}[e^{\log \frac{\mathbb{P}(\omega)}{\mathbb{Q}(\omega)}}] \tag{27}$$

$$= D_{KL}(\mathbb{P}||\mathbb{Q}) - \log \mathbb{E}_{\mathbb{Q}}\left[\frac{\mathbb{P}(\omega)}{\mathbb{Q}(\omega)}\right] \tag{28}$$

$$= D_{KL}(\mathbb{P}||\mathbb{Q}) - \log \int_\Omega \mathbb{Q}(\omega) \frac{\mathbb{P}(\omega)}{\mathbb{Q}(\omega)} d\omega \tag{29}$$

$$= D_{KL}(\mathbb{P}||\mathbb{Q}) - \log \int_\Omega \mathbb{P}(\omega) d\omega \tag{30}$$

$$= D_{KL}(\mathbb{P}||\mathbb{Q}) - \log 1 \tag{31}$$

$$= D_{KL}(\mathbb{P}||\mathbb{Q}) \tag{32}$$

Lemma 2. *For any function $T : \Omega \to \mathbb{R}$, the following*

equality holds:

$$D_{KL}(\mathbb{P}||\mathbb{Q}) \geq \mathbb{E}_{\mathbb{P}}[T] - \log \mathbb{E}_{\mathbb{Q}}[e^T] \tag{33}$$

Proof. We define the probability density function $\mathbb{G}$ as:

$$\mathbb{G}(\omega) \triangleq \frac{\mathbb{Q}(\omega)e^T}{\mathbb{E}_{\mathbb{Q}}[e^T]} \tag{34}$$

Note that $\mathbb{G}$ satisfies the non-negativity and the integral of its probability density function (PDF) over the input space equals 1:

$$\int_{\Omega} \mathbb{G}(\omega)d\omega = \int_{\Omega} \frac{\mathbb{Q}(\omega)e^T}{\mathbb{E}_{\mathbb{Q}}[e^T]}d\omega = \int_{\Omega} \frac{\mathbb{E}_{\mathbb{Q}}[e^T]}{\mathbb{E}_{\mathbb{Q}}[e^T]}d\omega = 1 \tag{35}$$

Then, we calculate the difference between the two sides of 33 to obtain:

$$\begin{aligned}
&D_{KL}(\mathbb{P}||\mathbb{Q}) - \mathbb{E}_{\mathbb{P}}[T] + \log \mathbb{E}_{\mathbb{Q}}[e^T] \\
&= \mathbb{E}_{\mathbb{P}}\left[\log \frac{\mathbb{P}(\omega)}{\mathbb{Q}(\omega)} - T\right] + \log \mathbb{E}_{\mathbb{Q}}[e^T] && (36)\\
&= \mathbb{E}_{\mathbb{P}}\left[\log \frac{\mathbb{P}(\omega)}{\mathbb{Q}(\omega)e^T} + \log \mathbb{E}_{\mathbb{Q}}[e^T]\right] && (37)\\
&= \mathbb{E}_{\mathbb{P}}\left[\log \frac{\mathbb{P}(\omega)\mathbb{E}_{\mathbb{Q}}[e^T]}{\mathbb{Q}(\omega)e^T}\right] && (38)\\
&= \mathbb{E}_{\mathbb{P}}\left[\log \frac{\mathbb{P}(\omega)}{\mathbb{G}(\omega)}\right] && (39)\\
&= D_{KL}(\mathbb{P}||\mathbb{G}) \geq 0 && (40)
\end{aligned}$$

Based on the Lemma 1 and Lemma 2, we show that by choosing $T^* = \log \frac{\mathbb{P}}{\mathbb{Q}}$, we obtain:

$$D_{KL}(\mathbb{P}||\mathbb{Q}) = \mathbb{E}_{\mathbb{P}}[T^*] - \log \mathbb{E}_{\mathbb{Q}}[e^{T^*}] \tag{41}$$

Additionally, for any function $T : \Omega \to \mathbb{R}$,

$$D_{KL}(\mathbb{P}||\mathbb{Q}) \geq \mathbb{E}_{\mathbb{P}}[T] - \log \mathbb{E}_{\mathbb{Q}}[e^T] \tag{42}$$

holds. Hence,

$$D_{KL}(\mathbb{P}||\mathbb{Q}) = \sup_{T:\Omega\to\mathbb{R}} \mathbb{E}_{\mathbb{P}}[T] - \log \mathbb{E}_{\mathbb{Q}}[e^T], \tag{43}$$

6.9. Implementation Details

The trajectory denoise model $\Phi_{\mathbf{TDM}}$ is implemented using a 3-layer Transformer with a feature dimension of 256 and the attention head is set to 4. The number of masked locations is set to 2 in our experiments. We empirically set the trade-off parameter β to 0.01 and the margin Δ to 0.05. Additionally, we set the trade-off parameters α, δ, and γ to 0.01, 1 and 0.01, respectively. For the Wavelet denoising method, we utilize the Daubechies wavelet to decompose the signals, and the level is set to 2. We employ the soft-threshold method, with a threshold value set to 0.2. Regarding the EMA method, we empirically determine the weighted parameter to be 0.75. All experiments are conducted on the PyTorch platform with 4 NVIDIA RTX3090 GPUs.

Reference

[1] Krull, Alexander, Tim-Oliver Buchholz, and Florian Jug. Noise2void-learning denoising from single noisy images. *In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp.2129-2137, 2019.