APPENDIX

A ANONYMIZED CODE

For an anonymized version of our code, please see: anonymous.4open.science/r/polysona

B WEIGHT-MIXING IMPLEMENTATION

PyTorch-Style Forward Pass of Weight-Mixing Mixture-of-LoRA Layer

```
expert_alphas = router(...) # (b, num_experts)

expert_weight_A = (expert_alphas[..., None, None] * self.
    expert_weight_A[None]).sum(
     dim=1
)  # (b, r, in_features)
expert_weight_B = (expert_alphas[..., None, None] * self.
    expert_weight_B[None]).sum(
     dim=1
)  # (b, out_features, r)

output = torch.einsum(
     "bi,bri->br", x, expert_weight_A
)  # (b, in_features) @ (b, r, in_features) -> (b, r)
output = torch.einsum(
     "br,bor->bo", output, expert_weight_B
)  # (b, r) @ (b, out_features, r) -> (b, out_features)
output = self.dropout(output)  # (b, out_features)

output = F.linear(x, w0, b0) + output  # w0x + BAx + b0
```

C Qualitative Results on Argoverse

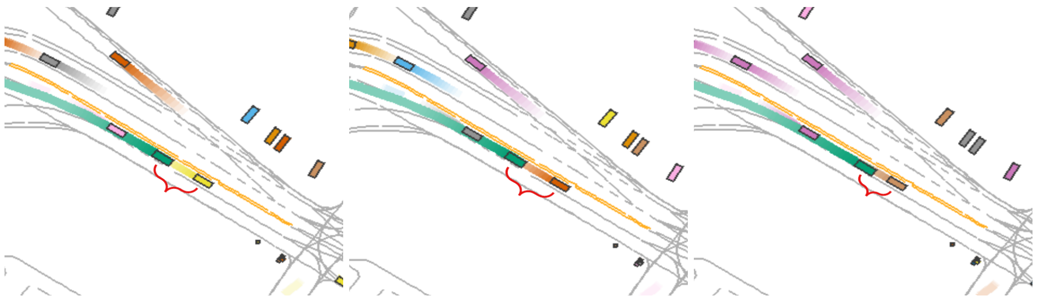


Figure 5: **Qualitative result: Rollouts on a lane change scenario with different experts on Ours+CAT.** We show how different experts behave under the same lane change scenario, where the goal is to lane change between two agents. While rollouts from Expert 1 and 2 are very similar (maintaining at least one vehicle length from the leading vehicle), Expert 3, which has the lowest prior probability, predicts a trajectory with much smaller headway distance. Animations of these scenarios are better visualized on our project website, linked in the abstract.

D Style Consistency Metric Visualization

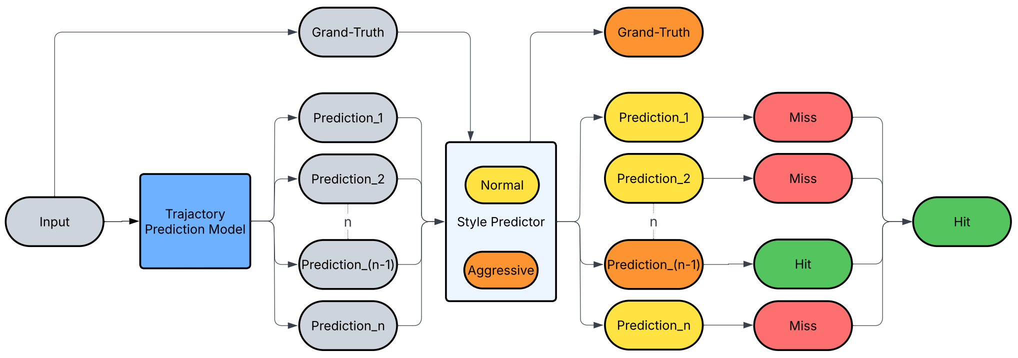


Figure 6: **Style Consistency Metric.** Given a driving scenario ("Input"), the black-box trajectory predictor generates n candidate futures ($\text{Prediction}_1, \ldots, \text{Prediction}_n$). A learned style predictor then assigns each candidate—and the true future ("Ground-Truth")—to one of two clusters (e.g. "Normal" vs. "Aggressive"). If at least one of the n predicted trajectories shares the same style label as the ground-truth, the sample is marked a *Hit*; otherwise it is a *Miss*. This hit/miss outcome directly measures whether the model's multi-modal outputs *cover* the driver's actual style, beyond conventional displacement errors.

E Style Consistency Metric

To explicitly measure a model's ability to cover the correct driving style, as shown in Figure 6, we propose the *Style Miss Rate* a style consistency metric based on kinematic clustering following the clustering methodology introduced in (Zheng et al., 2025), and a hit/miss criterion:

Extract kinematic static: For each trajectory τ, compute a feature vector

$$\phi(\tau) = \left[\max_t \left|a(t)\right|,\ Var(a),\ Var(v),\ \gamma\right]^\top \in \mathbb{R}^d, \tag{8}$$

where a is acceleration (with $\max_t |a(t)|$ denoting the peak absolute acceleration over the trajectory, i.e. the highest instantaneous acceleration magnitude), v is speed, and

$$\gamma = \frac{\mathrm{Var}\big(j(t)\big)}{\mathbb{E}\big[j(t)\big]}$$

is the jerk-variance ratio as defined in Murphey et al. (2009).

Learn style clusters: Fit a Gaussian Mixture Model (GMM) with $k = 2$ on the set of all ground-truth kinematic embeddings in the evaluation set $\{\phi(\tau_n^*)\}_{n=1}^{N}$, yielding a cluster assignment function

$$C(\phi) \in \{\texttt{"normal"}, \texttt{"aggressive"}\}. \tag{9}$$

Normal and aggressive are assigned based on the mean speed of each cluster. A cluster with higher mean speed will be assigned as aggressive.

Assign styles to predictions: For each sample n, let $\{\hat{\tau}_{n,i}\}_{i=1}^{6}$ be the six predicted trajectories. Define

$$s_n^* = C\big(\phi(\tau_n^*)\big), \quad s_{n,i} = C\big(\phi(\hat{\tau}_{n,i})\big). \tag{10}$$

Define hit/miss: A *hit* occurs if at least one predicted style matches the ground-truth style:

$$\mathrm{Hit}_n = \{\exists\, i \;:\; s_{n,i} = s_n^*\}, \quad \mathrm{Miss}_n = 1 - Hit_n. \tag{11}$$

Style Miss Rate: The overall metric is

$$\mathrm{SMR} = \frac{1}{N} \sum_{n=1}^{N} \mathrm{Miss}_n. \tag{12}$$

By construction, the SMR goes beyond pure spatial accuracy: it measures whether the model "covers" the driver's true style among its multi-modal outputs. A style-agnostic predictor may achieve low ADE/FDE by clustering its modes around average behavior, but will incur a high miss rate on aggressive samples. In contrast, a style-aware model—conditioned on inferred driving-style embeddings—should include at least one candidate trajectory whose kinematics align with the true style, yielding a lower SMR.

F Style Miss Rate Evaluation on Ablation Variants

To further assess how each component of our mixture-of-experts framework contributes to style coverage, we compute the Style Miss Rate (SMR) on the same ablation variants presented in Table 4. That is, for each model variant—removing reconstruction, KL loss, entropy regularization, etc.—we evaluate how often none of its multi-modal predictions match the true driving style cluster. The resulting SMR values are reported in Table 6. This analysis shows that the ablations which most degrade traditional error metrics (e.g. reconstruction and KL removal) also incur the largest increases in SMR, indicating a direct link between component contributions and the model's ability to cover the driver's style.

Table 6: **Style Miss Rate (SMR) for each ablation variant.**

Ours	SMR$\downarrow$
+LinearRouter	0.2246
+Full Finetuning	0.2033
-Social Forces	0.2229
-Context Features	0.2236
-KL Loss	0.2267
-Reconstruction	0.2263
-Expert Entropy Loss	0.2260

G How does the model reason? Taking a look at the saliency maps.

To gain insight into which input features the model attends when forecasting agent trajectories, we compute saliency maps by measuring the sensitivity of the most likely predicted trajectory with respect to each input feature. Formally, let $X = \{A_{\text{in}}, M_{\text{in}}\}$ denote the concatenation of historical object trajectories A_{in} and map polylines M_{in}. If $\hat{p}_\tau$ is the probability of trajectory τ, then let $\hat{\tau}^* = \arg\max_\tau \hat{p}_\tau$ be the most likely predicted trajectory after a forward pass from a model. Then, we compute the gradient

$$\nabla_X \hat{\tau}^* = \frac{\partial \hat{\tau}^*}{\partial X}$$

via back-propagation, and form the saliency map

$$S(X) = \log(\|\nabla_X \hat{\tau}^*\|_2 + 1).$$

We use logarithmic scaling above to better display nuances in smaller gradient magnitudes. For visualizing this saliency map, we render the map polylines and the agents' historical trajectories colored using $S(X)$ on the "jet" color scheme (dark blue to green to dark red). Warmer colors highlight map segments or agent trajectories that the model deems more important for predicting future trajectories. Likewise, lighter colors highlight areas of less importance. Each agent vehicle is also colored on the same scale based on the maximum saliency value of its historical trajectory. We generate this visualization for MTR+Actions, Ours+CAT, and Ours+MoV:

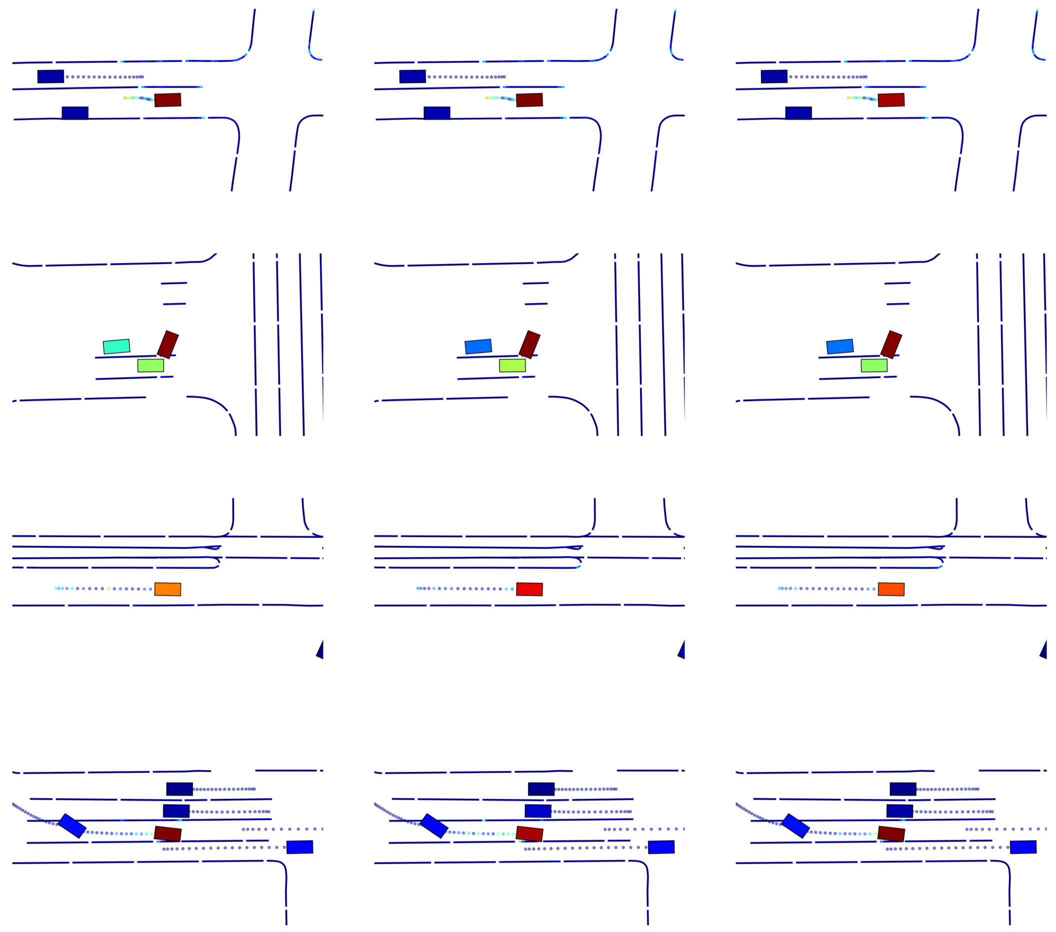


Figure 7: **Visualization of scenario feature saliency (Continued on the next page).** Saliency maps are visualized for 10 randomly selected scenarios from Argoverse on MTR+Actions (Left), Ours+CAT (Middle), and Ours+MoV (Right).

.

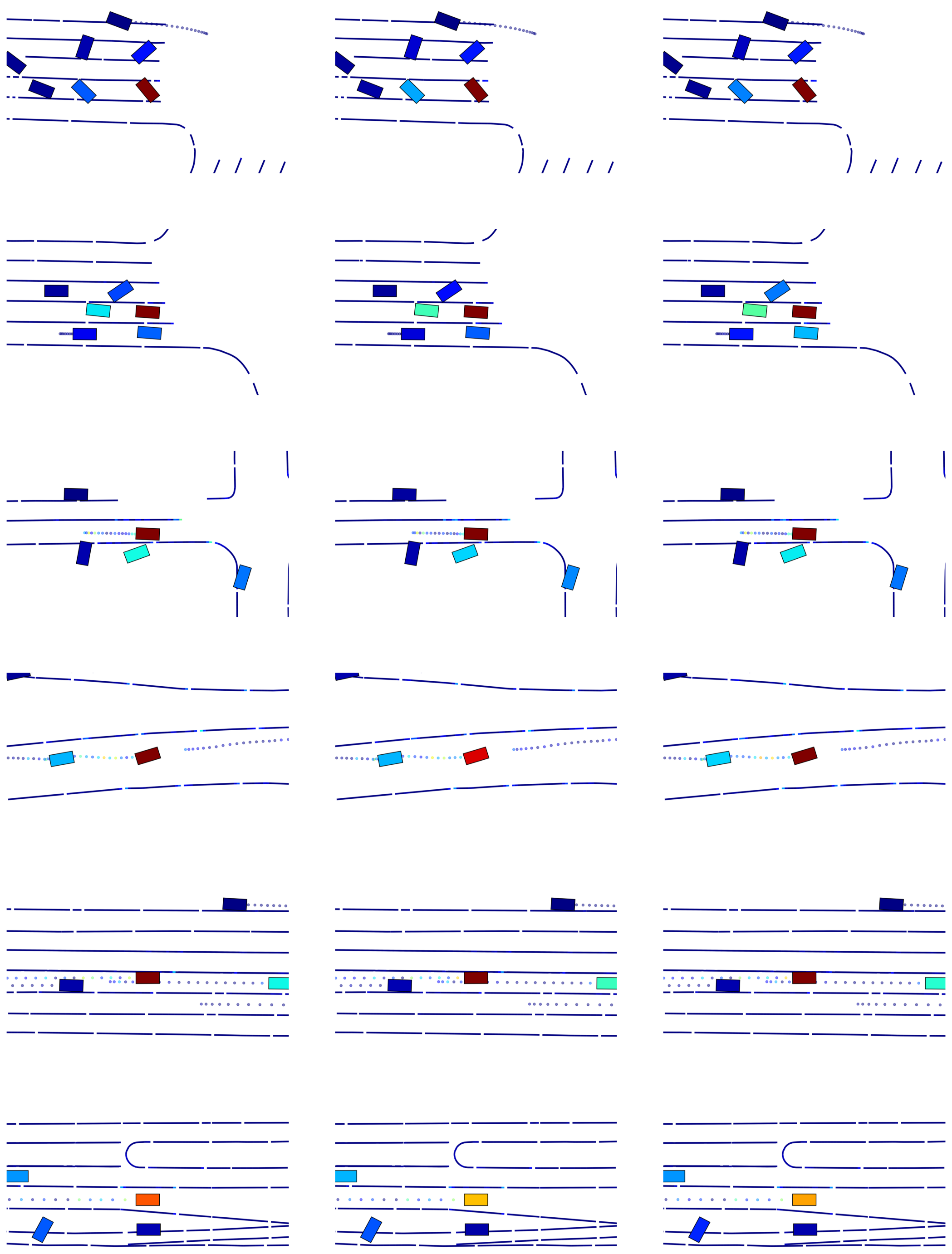
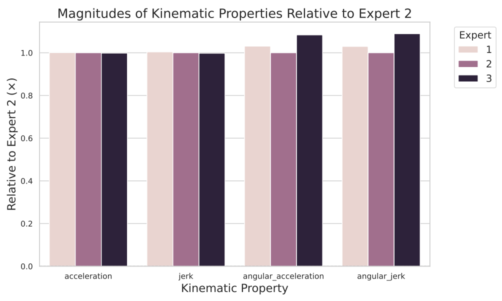
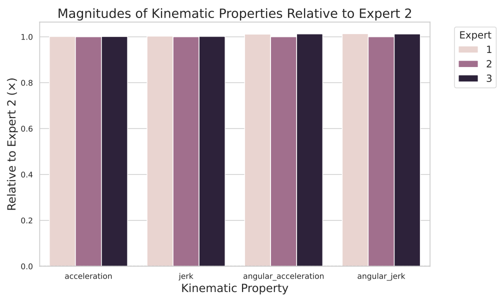


Figure 7: **Visualization of scenario feature saliency.** Saliency maps are visualized for 10 randomly selected scenarios from Argoverse on MTR+Actions (Left), Ours+CAT (Middle), and Ours+MoV (Right). The top 2 scenarios feature mostly stationary agents. In each scenario and model, the ego vehicle has the most saliency (colored red), followed by nearby agents and map polylines.

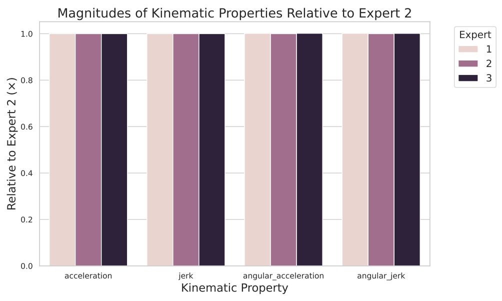
H ADDITIONAL VISUALIZATIONS FOR RELATIVE KINEMATICS BY EXPERT



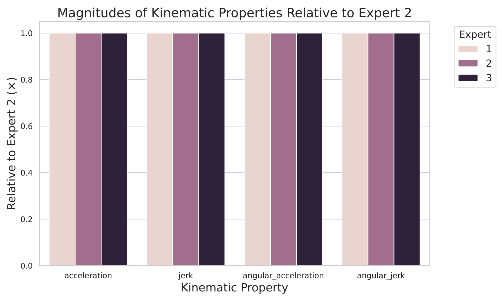
(a) C-Poly



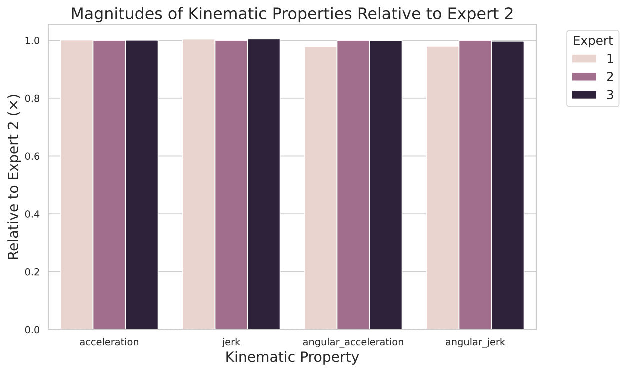
(b) HyperFormer



(c) Polytropon



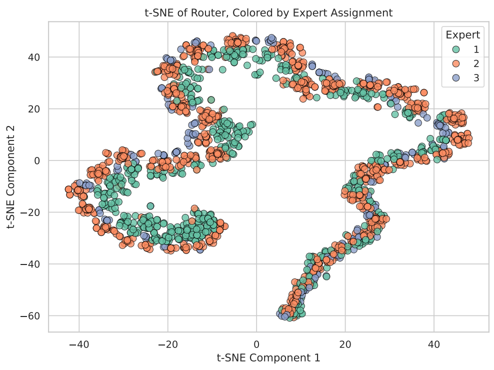
(d) MoV



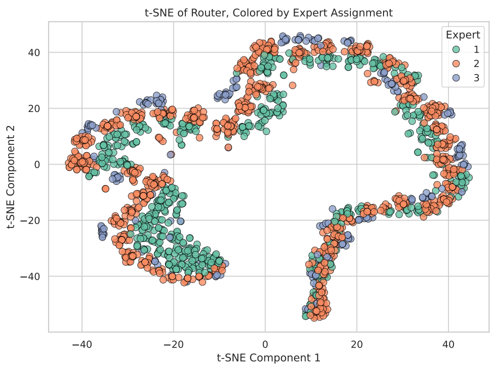
(e) CAT

Figure 8: **Kinematic magnitude comparisons for all variants of our approach.** Experiments run with seed 0 are plotted.

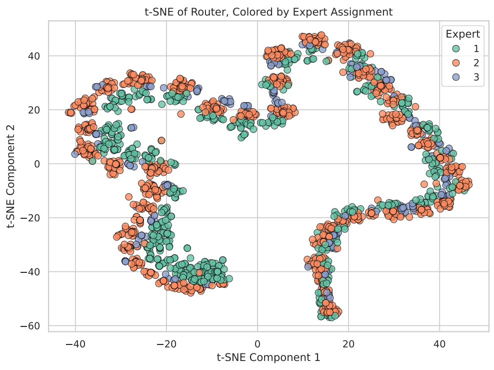
I Additional t-SNE plots for Router Embeddings by Expert



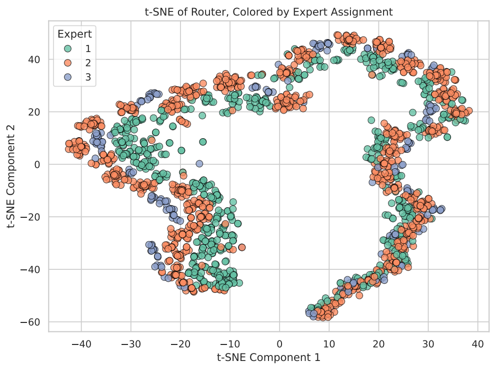
(a) C-Poly



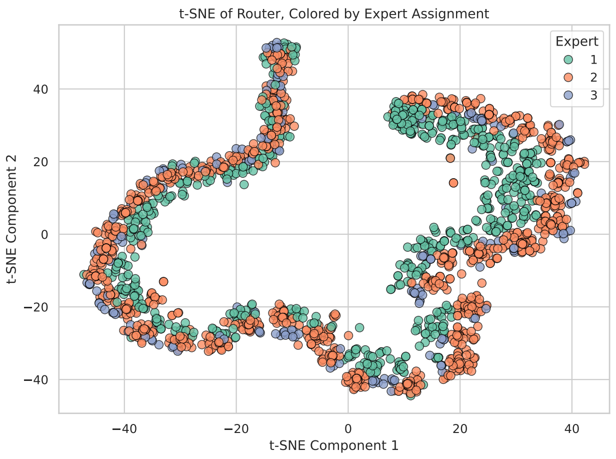
(b) HyperFormer



(c) Polytropon



(d) MoV



(e) CAT

Figure 9: **t-SNE visualization of router embeddings for all variants of our approach.** Experiments run with seed 0 are plotted.

J Baseline MTR Model Hyperparameters

Table 7: Hyperparameters used for training the MTR baseline model.

Hyperparameter	**Value**
Context Encoder	
NAME	MTREncoder
NUM_OF_ATTN_NEIGHBORS	7
NUM_INPUT_ATTR_AGENT	39
NUM_INPUT_ATTR_MAP	29
NUM_CHANNEL_IN_MLP_AGENT	256
NUM_CHANNEL_IN_MLP_MAP	64
NUM_LAYER_IN_MLP_AGENT	3
NUM_LAYER_IN_MLP_MAP	5
NUM_LAYER_IN_PRE_MLP_MAP	3
D_MODEL	256
NUM_ATTN_LAYERS	6
NUM_ATTN_HEAD	8
DROPOUT_OF_ATTN	0.1
USE_LOCAL_ATTN	True
Motion Decoder	
NAME	MTRDecoder
NUM_MOTION_MODES	6
D_MODEL	512
NUM_DECODER_LAYERS	6
NUM_ATTN_HEAD	8
MAP_D_MODEL	256
DROPOUT_OF_ATTN	0.1
NUM_BASE_MAP_POLYLINES	256
NUM_WAYPOINT_MAP_POLYLINES	128
LOSS_WEIGHTS.cls	1.0
LOSS_WEIGHTS.reg	1.0
LOSS_WEIGHTS.vel	0.5
NMS_DIST_THRESH	2.5
Training	
max_epochs	40
learning_rate	0.0001
learning_rate_sched	[22, 24, 26, 28]
optimizer	AdamW
scheduler	lambdaLR
grad_clip_norm	1000.0
weight_decay	0.01
lr_decay	0.5
lr_clip	0.000001
WEIGHT_DECAY	0.01
train_batch_size	64
eval_batch_size	64
Data	
max_num_agents	64
map_range	100
max_num_roads	768
max_points_per_lane	20
manually_split_lane	True
point_sampled_interval	1
num_points_each_polyline	20
vector_break_dist_thresh	1.0
predict_actions	True

K PolySona Model Hyperparameters

Table 8: Hyperparameters used for training the PolySona models. Rows highlighted in yellow indicate differences from the baseline MTR configuration.

Hyperparameter	Value
Context Encoder	
NAME	MTREncoder
NUM_OF_ATTN_NEIGHBORS	7
NUM_INPUT_ATTR_AGENT	39
NUM_INPUT_ATTR_MAP	29
NUM_CHANNEL_IN_MLP_AGENT	256
NUM_CHANNEL_IN_MLP_MAP	64
NUM_LAYER_IN_MLP_AGENT	3
NUM_LAYER_IN_MLP_MAP	5
NUM_LAYER_IN_PRE_MLP_MAP	3
D_MODEL	256
NUM_ATTN_LAYERS	6
NUM_ATTN_HEAD	8
DROPOUT_OF_ATTN	0.1
USE_LOCAL_ATTN	True
Motion Decoder	
NAME	PolySonaDecoder
NUM_MOTION_MODES	6
INTENTION_POINTS_FILE	`cluster_64_center_dict_6s.pkl`
D_MODEL	512
NUM_DECODER_LAYERS	6
NUM_ATTN_HEAD	8
MAP_D_MODEL	256
DROPOUT_OF_ATTN	0.1
NUM_BASE_MAP_POLYLINES	256
NUM_WAYPOINT_MAP_POLYLINES	128
LOSS_WEIGHTS.cls	1.0
LOSS_WEIGHTS.reg	1.0
LOSS_WEIGHTS.vel	0.5
NMS_DIST_THRESH	1.0
Training	
max_epochs	10
learning_rate	0.001
learning_rate_sched	[22, 24, 26, 28]
optimizer	AdamW
scheduler	polynomialLR (power=2)
grad_clip_norm	1000.0
weight_decay	0.00
lr_decay	0.5
lr_clip	0.000001
train_batch_size	256
eval_batch_size	256
predict_actions	True
lora_rank	4
freeze_encoder	True
freeze_decoder	True
attention_only	False
num_personas	3
prior	[0.3, 0.6, 0.1]
λ_{recon}	50
λ_{KL}	50
λ_{entropy}	25
seed	0 / 1 / 2
Data	
max_num_agents	64
map_range	100
max_num_roads	768
max_points_per_lane	20
manually_split_lane	True
point_sampled_interval	1
num_points_each_polyline	20
vector_break_dist_thresh	1.0

L Impact of Rank on Performance

Table 9: **Comparison of Ours+CAT Across Different Ranks.**

Rank	brierFDE↓	minADE↓	minFDE↓	MissRate↓
2	2.1593	0.8573	1.7059	0.3151
4	2.1607	0.8624	1.7041	0.3171
8	2.1578	0.8578	1.7042	0.3120
16	2.1668	0.8610	1.7102	0.3151

Table 10: **Comparison of Ours+CAT Across Different Ranks, Grouped by Kalman Difficulty and TDBM Driving Styles.**

	Kalman Difficulty			TDBM Driving Styles			
Rank	Easy	Medium	Hard	Timid	Careful	Reckless	Threatening
2	0.8120	1.1675	3.9875	0.8903	0.8865	0.8577	0.8172
4	0.8188	1.1678	2.4978	0.8793	0.8477	0.8655	0.8161
8	0.8130	1.1641	3.9882	0.8913	0.9833	0.8581	0.8183
16	0.8149	1.1758	4.2696	0.8944	0.8858	0.8613	0.8225

M STANDARD DEVIATION TABLE

Table 11: **Standard Deviation of Trajectory Prediction Benchmark Performance Comparisons.**

Method	brierFDE↓	minADE↓	minFDE↓	MissRate↓
Ours+Polytropon Ponti et al. (2023)	0.0004	0.0004	0.0004	0.0009
Ours+C-Poly Wang et al. (2024a)	0.0026	0.0011	0.0025	0.0003
Ours+HyperFormer Karimi Mahabadi et al. (2021)	0.0017	0.0004	0.0017	0.0010
Ours+CAT Prabhakar et al. (2024)	0.0019	0.0012	0.0018	0.0010
Ours+MoV Zadouri et al. (2024)	0.0010	0.0007	0.0010	0.0004

Table 12: **Standard Deviation of minADE Comparison by Kalman Difficulty and TDBM Driving Style groups.**

	Kalman Difficulty			TDBM Driving Styles			
Method	Easy	Medium	Hard	Timid	Careful	Reckless	Threatening
Ours+PolyTropon Zadouri et al. (2024)	0.0003	0.0032	0.0040	0.0005	0.0038	0.0004	0.0005
Ours+C-Poly Wang et al. (2024a)	0.0013	0.0003	0.0146	0.0009	0.0286	0.0012	0.0011
Ours+HyperFormer Karimi Mahabadi et al. (2021)	0.0006	0.0013	0.0060	0.0006	0.0330	0.0004	0.0007
Ours+CAT Prabhakar et al. (2024)	0.0012	0.0051	0.0682	0.0014	0.0340	0.0012	0.0012
Ours+MoV Zadouri et al. (2024)	0.0007	0.0006	0.0041	0.0008	0.0026	0.0007	0.0006

Table 13: Hyperparameter sweep of # of classes used in PolySona training.

# Latent Classes	Easy / minADE	Medium / minADE	Hard / minADE	Overall minADE
2	**0.8108**	1.1951	3.5275	**0.8589**
3	0.8188	**1.1678**	**2.4978**	0.8624
4	0.8119	1.1990	3.5116	0.8603
5	0.8120	1.1996	3.5191	0.8605
6	0.8109	1.1986	3.5729	0.8595
8	0.8123	1.2027	3.4217	0.8610
10	0.8122	1.2007	3.6048	0.8609

N Hyperparameter Sweet: Number of Latent Classes Modeled