Polysona: Modular Driving Styles in Trajectory Prediction

Laura Zheng[1], Hamidreza Yaghoubi Araghi[1], Tony Wu[1], Tianyi Zhou[2], and Ming C. Lin[1]

***Abstract*— In rare but safety-critical driving scenarios, we hypothesize that trajectory outcomes become increasingly multi-modal based on differences between driver style compared to non-critical, common scenarios. However, current approaches for trajectory prediction rarely account for differences in driving style, which may lead to "averaged" driving style in predictions. While average-case behavior may work well in straight driving, easy scenarios, it limits the diversity of outcomes in more complex scenes or in rare events. Extraction of driving style has several benefits, as it enables simulation of counterfactual outcomes in real-world log replays and potentially more accurate predictions through style-consistent predictions. In this paper, we present *Polysona*, a parameter-efficient Mixture-of-Experts framework for extraction of latent driving styles in trajectory prediction models. We choose a parameter-efficient approach to reduce forgetting in well-generalized trajectory prediction models, while offering portability of trained driving style modules. In our results, we benchmark different mixture-of-LoRA approaches with our method and provide qualitative analysis on how the learned experts specialize. Additional qualitative results can be found on our project website: `https://laurayuzheng.github.io/polysona/`**

I. Introduction

As autonomous driving becomes an increasingly accessible technology, handling *mixed autonomy* traffic systems will also become an increasingly important research question. Mixed autonomy traffic systems comprise of both human drivers and autonomous drivers, to varying degrees. The general autonomous driving stack approaches driving in sequential modules, each responsible for a specific task: perception, prediction, planning, and control. Much of the difficulty in trajectory prediction lies not in the common cases such as lane following and sparser suburban roads, but rather the complex cases with many stressors involved. With more stressors, human driving behavior diverges into a splay of different outcomes; this is expected due to the General Adaptation Syndrome, a spectrum of "fight or flight" responses in humans which has also been shown to influence crowd navigation behavior of humans [1].

Currently, this variation in human decision making of other human drivers on the road has not been modeled in trajectory prediction frameworks of autonomous vehicles. Yet, this variable would influence multi-modal outcomes in cases, where an arbitrary decision produces drastically different trajectories, even with a fixed traffic context and route intent. We hypothesize that *driving style* is increasingly impactful during these rare events, where there are typically more stressors pressuring drivers to make decisions in a split second. One motivating example is the case where the driver needs to change lanes to achieve their route goals, but is directly blocked by other drivers in the target lane. The driver must then choose between two specific outcomes: decelerate and change lanes behind the other drivers, or accelerate and change lanes in front of other drivers. In terms of common trajectory prediction metrics, the difference between the two very plausible outcomes is large; one decision would be penalized, while the other's likelihood would be increased with respect to model parameters. We illustrate this example in Figure 1, where the green vehicle's possible trajectory paths diverge greatly per timestep.

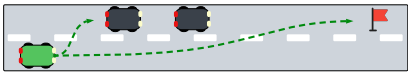


Fig. 1: **Motivating Example: Lane change occupied by other vehicles.** The ego vehicle has the goal of changing lanes and reaching a particular waypoint. At its current velocity, a direct lane change would result in a collision with the other vehicles. Here, the ego vehicle is faced with a natural stressor to make a decision: either decelerate and merge behind the other vehicles, or accelerate and pass the other vehicles first, then merge. Both outcomes are plausible, yet lead to drastically different trajectories by average displacement error metrics, which are computed per-timestep.

Our approach, *Polysona*, models latent driving style as a parameter-efficient Mixture-of-Experts, where we train several expert Low-Rank Adapters (LoRA) [2] guided by real world priors on driving style. To guide learned experts towards representations related to driving style, we use a vehicle traffic dynamics adaptation to the Social Forces Model (SFM) [3] for expert routing. Originally introduced to model pedestrian dynamics, the SFM represents interactions among agents as attractive and repulsive forces, and has since been adapted for robotics tasks such as social-force-based robot navigation among humans [4], making it a natural, interpretable signal for characterizing interaction pressure among multiple moving agents (i.e., cars). We motivate the use of parameter-efficient paradigms for its efficiency and preservation of baseline performance. Additionally, adapters make it easy to control for expert behavior, which opens possibilities for counterfactual simulation outcomes.

Our main contributions are summarized as follows:

1) A parameter-efficient, simple yet effective Mixture-of-LoRA (MoL) approach for latent modeling of driving style in trajectory prediction (Fig. 2).
2) A router design guided by social force features from

[1]University of Maryland, College Park, MD, USA
[2]Mohamed bin Zayed University of Artificial Intelligence (MBZUAI), Abu Dhabi, UAE

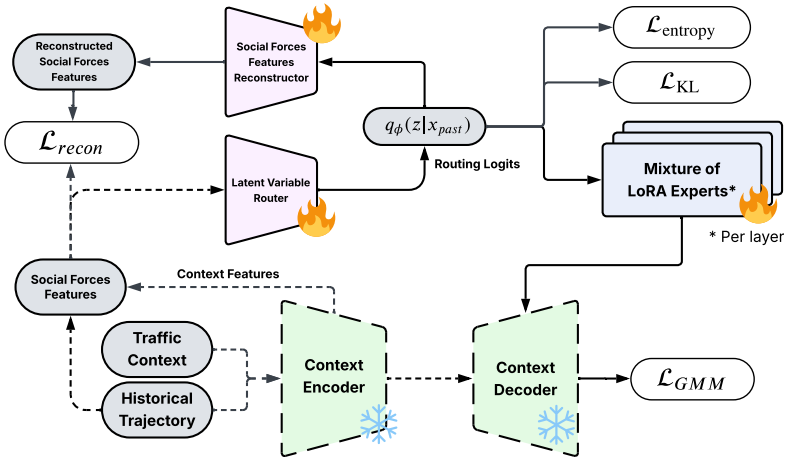


Fig. 2: **Training Overview.** We use a Mixture-of-LoRA (MoL) approach to learn a driving style latent variable for trajectory prediction models. Solid lines indicate gradient flow in backpropagation. In summary, we construct social forces features from historical agent trajectories and train a routing network in a Variational Autoencoder-like fashion; in other words, the router predicts logits for discrete latent classes, and a reconstructor network then learns to re-construct social forces features. The predicted routing logits and then used to select experts in the MoL layers, which are attached to certain decoder layers of the base model. All modules except the base model are trained end-to-end.

traffic scenarios;

3) Qualitative analysis on expert specialization with our MoL approach for latent variable modeling.

II. Related Works

A. Driving Style Modeling in Autonomous Driving

Driving style can be useful for several modules in the autonomous driving stack. For policy training, style modeling can be used for personalization of driving policies to maximize rider comfort, especially if the policy is expected to mimic the human's own driving [5], [6]. On the other hand, driving style can also be useful for planning, where accurately predicted human driving behavior is essential. In trajectory forecasting, some work has investigated driving styles for specific maneuvers, such as lane changes [7], [8], or specific context such as highways [9] and intersection [10] scenarios. Several efforts model driving style from observable trajectory features, such as with Graph Neural Networks (GNNs) [11] or classical clustering based on kinematic properties [9]. Most recent efforts use deep learning, with either recurrent networks [7] or generative models [12] to extract driving style features implicitly. One common theme is that driving style is generally considered a latent variable modeled as a function of kinematic properties such as acceleration and jerk. However, many of these works require an annotated dataset [7] or a specialized backbone architecture [12], [11].

In our work, we distinguish our goals from previous works in that we want to complement and maintain performance from state-of-the-art trajectory prediction models, whilst also extracting generalized representations of driving style.

B. Parameter-Efficient Mixture of Experts for Specialization in Driving

Parameter-efficient Mixture-of-Experts (PE-MoE) is a key focus in language modeling and image diffusion, where efficient fine-tuning atop foundation models is essential. Accordingly, prior PE-MoE work has largely centered on language and diffusion tasks. We extend these paradigms to trajectory prediction, aiming to extract a latent variable that captures driving style. PE-MoEs typically use collections of LoRA adapters as experts, which can encode specialized concepts in image diffusion [13], alleviate catastrophic forgetting [14], and match the performance of full fine-tuning at lower cost in language tasks [15]. Using multiple pre-trained LoRAs also enables reusing task-specific skills. Multi-task learning with LoRAs similarly targets a latent skill representation (task-skill matrix), reusing skills across tasks. Notable examples include Polytropon [16], C-Poly [17], and Hyperformer [18]. These are complementary to our work, and our MoE design is compatible with most.

Some works apply parameter-efficient fine-tuning to trajectory prediction. Forecast-PEFT [19] uses prompt vectors and LoRA layers to achieve results close to full fine-tuning. Our use case is different; we learn a latent variable across different driving styles in a parameter-efficient manner. In robotics, MoE architectures have been used for vehicle trajectory prediction, though focused on learning intentions or goals [20]; we instead learn driving style, where the goal is fixed and we model different ways to achieve it. Alternative latent-variable methods like VAEs [21] are more computationally costly and limit latent portability across models. Another discrete latent approach, VQ-VAE [22], learns codebook embeddings via nearest-neighbor lookup. However, VQ-VAE is prone to codebook collapse when the number of codes is small, as in our setting (E=3), and requires a dedicated encoder to map inputs into the codebook embedding space, which conflicts with a parameter-efficient design. Since our approach already employs discrete categorical latents via Gumbel-Softmax, VQ-VAE would be

a lateral alternative rather than a revealing comparison; the more informative axis is discrete (ours) versus continuous (VAE) latent representations.

III. Methodology

In this section, we discuss our latent variable modeling framework for driving style.

A. Latent Driving Styles with Experts

Driving styles are difficult to model because the notion of "driving style" is ill-defined and non-standardized, especially in autonomous driving research. Due to the difficulties of labeling driving style consistency and reliably, many datasets often do not include driving style related information, making this problem challenging—akin to latent variable modeling. In this case, we observe *driving style* as an outcome of latent variable modeling.

In our approach, we make the key assumption that driving style must be strongly correlated to observed second-order kinematics. Second-order kinematics is directly related to the actions a driver takes with the steering wheel, throttle, and brake, as input to control a vehicle. We illustrate our approach in Figure 2. Our approach first assumes a pre-trained trajectory forecasting model based on encoder-decoder transformer architectures. This model is frozen and serves as a shared representation of general driving behavior. Each MoL layer, depicted in blue, combines expert adapter weights according to the routing weights from the global router.

Mixture of LoRA Experts. *Low-rank adapters (LoRA)* [2] are a popular method for parameter-efficient finetuning (PEFT), which is popularly used for large language models. LoRA represents finetuning as a low-rank update to pre-trained weight matrices $W_0 \in \mathbb{R}^{d\times k}$:

$$h = W_0 x + \Delta W x + b_0 = W_0 x + BAx + b_0 \tag{1}$$

where $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times k}$, and the rank of both matrices $r \ll \min(d, k)$. During training, the pre-trained weights W_0 are frozen and only matrices B and A are updated.

Mixture-of-experts (MoE) is a modular paradigm which consists of N learned experts $\{E_1, ..., E_N\}$ and a router function which combines outputs from each expert in a weighted voting fashion. The experts are typically learned jointly and end-to-end, where the objective is for each expert to learn a specialized representation. The voting weights are known as *routing weights*, and a mixture function G determines how the outputs are combined using the routing weights. In our work, G is simply a weighted sum.

For our use case, we would like to learn specialized experts which encode latent driving styles, but also want to avoid the computational burden of full MoEs. Thus, we use Mixture-of-LoRA (MoL), which trains a set of expert LoRAs, instead of another trajectory prediction model. The mixing paradigm of the LoRA experts can be interchanged with any existing MoL paradigm; we benchmark several in our experiments. To the best of our knowledge, MoL has not yet been benchmarked in trajectory prediction, making our experiments among the first to apply parameter-efficient techniques to trajectory models.

Expert Routing based on Social Forces. We create a global expert router based on social forces and environment context features from the context encoder of the backbone network. The router is treated like a variational posterior estimator $q_\phi(z|x_{past})$ for social forces and context features; we sample the latent differentiably with Gumbel-Softmax from the predicted latent logits, then reconstruct the input social forces and context features. We compute social force features from the input trajectory information based on the Social Forces Model. For a particular intersection between vehicles i and j, the repulsion force between them is defined as:

$$F_{ij} = \alpha \cdot e^{(\delta - \|x_j - x_i\|)/\beta} \cdot \hat{\theta}_{ij} \tag{2}$$

where α is a hyperparameter scaling repulsion magnitude, β scales the decay rate as pairwise distances become larger, and δ corresponds to the preferred distance between agents. All three hyperparameters are set to 1 in our experiments. When constructing social forces features, interactions between a vehicle and itself are masked out, along with vehicles beyond a radius threshold of 10 meters and vehicles not traveling in the same direction. To combine social forces features with context embeddings of hidden dimension d, we concatenate both to a vector of dimension $2d$, then use a single dense layer to fuse features together, resulting in a final feature dimension of d. This fused representation is used as input to the global expert router.

The expert router architecture is a 2-layer MLP: a dense layer projecting to a hidden dimension of 64, a normalization layer, ReLU nonlinearity, dropout at p=0.1, and a dense layer outputting logits for E classes. We use E=3 experts following traffic psychology literature suggesting three distinct driver types [23].

Loss Objectives. We use four loss terms to guide the MoL layers towards a driving style representation. First, we maintain the original trajectory prediction objective, which maximizes the likelihood of the ground-truth trajectory given a predicted Gaussian Mixture of trajectory outcomes ($\mathcal{L}_{\text{GMM}}$). Second, since driving style is an unsupervised variable, we enforce a VAE-like objective on the global router, which eventually serves as a "persona classifier." We have two terms enforced on the router: a reconstruction term, which reconstructs the social forces features s, and a KL divergence term, which maximizes the Evidence Lower Bound (ELBO). We also include an entropy regularizer to encourage balanced expert utilization. Thus, the loss objective in training $\mathcal{L}$ becomes:

$$\mathcal{L} = \mathcal{L}_{\text{GMM}} + \lambda_{\text{recon}}\mathcal{L}_{\text{recon}} + \lambda_{KL}\mathcal{L}_{\text{KL}} + \lambda_{\text{entropy}}\mathcal{L}_{\text{entropy}} \quad (3)$$

$$\mathcal{L}_{\text{GMM}} = \mathbb{E}_{q_\phi}(z|x_{\text{past}})\left[-logp_\theta(x_{\text{future}}|z)\right] \quad (4)$$

$$\mathcal{L}_{\text{KL}} = KL(q_\phi(z|x_{\text{past}})\|p(z)) \quad (5)$$

$$\mathcal{L}_{\text{recon}} = \|s_{\text{orig}} - s_{\text{pred}}\|_2 \quad (6)$$

$$\mathcal{L}_{\text{entropy}} = -\sum_z q_\phi(z \mid x_{\text{past}}) \log q_\phi(z \mid x_{\text{past}}) \quad (7)$$

IV. Results

Hardware. Each experiment is trained on 32 GB memory, two AMD EPYC 7352 24-Core Processors, and two NVIDIA RTXA5000 GPUs.

Experiment setup. In our experiments, the base trajectory prediction model used is Motion Transformer (MTR), a state-of-the-art, open source, non-autoregressive approach. We train experiments on the Argoverse 2 dataset [24], which consists of ∼250,000 total scenarios. We use 183,333 samples for training and 22,979 samples for validation. Our experiments were conducted with the UniTraj framework [25] for trajectory prediction. In our experiments, the task is to predict the next future 6s trajectories, given 2s of agent history trajectories and road context polylines. Agent trajectories are sampled at 10Hz, for a total of 20 history frames and 60 future frames. All experiments are trained on 10 epochs and evaluated on the 10th epoch. All experiments are initialized from a trained MTR+Actions [26] checkpoint trained on Argoverse 2; thus, there is no need to account for domain shift in fine-tuning. All experiments are trained to learn three experts (or driving styles) with a prior probability of $[0.3, 0.6, 0.1]$ for each expert, respectively, which is inspired from driver distributions published by the NHTSA [23]. We swept the number of latent classes from 2 to 10 and found that three experts performed best empirically, particularly on medium and hard difficulty scenarios (see Appendix). In the base model, there are six decoder layers, where each decoder layer consists of agent and map attention blocks, as well as a Gaussian mixture prediction head. We apply Rank-4 MoL to the object attention and the Gaussian Mixture prediction head of the 3rd and 6th decoder layers (6 decoder layers total). This produces about 30 MoL layers in MTR experiments. The loss weights of Eq. (3) are set to $\lambda_{\text{recon}} = 50$, $\lambda_{\text{KL}} = 50$, and $\lambda_{\text{entropy}} = 25$ in all experiments. More hyperparameter details for experiment reproduction can be found in the Appendix.

Metrics. We report the following metrics (all ↓):

- **brierFDE**: Displacement error (m) of each prediction mode of a Gaussian Mixture, weighted by the mixture score. Accounts for both trajectory accuracy and model confidence.
- **minADE**: Minimum average displacement error (m) from the ground truth trajectory across predicted modes, averaged over time.
- **minFDE**: Minimum final displacement error (m) between the closest ending position of a predicted trajectory and the ground truth.
- **Miss Rate**: The fraction of samples where minFDE exceeds 2 meters.

A. Benchmark Comparisons

In Table I, we compare trajectory prediction performance against existing architectures. We also include TAE [27], a behavior-aware adversarial autoencoder that models aggressiveness and intention as latent variables for trajectory prediction and generation. TAE is the closest related work that explicitly models driving behavior in the latent space. We additionally compare against a CVAE [31] baseline, which replaces our discrete categorical routing and LoRA experts with a continuous Gaussian latent variable and FiLM conditioning [32] on the same frozen MTR backbone. With approximately 99.5K trainable parameters, this baseline isolates whether the discrete expert structure of our approach is necessary, or whether a simpler continuous latent suffices for style-conditioned prediction.

In Table II, we compare different MoL schemes applied in our framework; each MoL experiment is averaged over three fixed random seed runs. C-Poly [17], HyperFormer [18], and Polytropon [16] are parameter-efficient approaches from multi-task learning (MTL) literature, rather than from Mixture-of-LoRA. While MTL is typically distinct from MoL work, these MTL methods are very similar in that they use distinct LoRA matrices per task. However, differently from our use case, they 1) assume that LoRA adapters are pre-trained and 2) assume task classes are a given. This means that the router cannot be learned end-to-end with the experts. For MTL experiments, the router only depends on VAE objectives to choose experts, since gradients from the GMM loss do not propagate back to the router (∇Router in Table II). Overall, all variants of our method improve both overall performance metrics and across most scenario groups detailed in Tables III and V. Amongst different MoL paradigms, we find that MoV [15] achieves the best all-around performance boosts. As an additional plus, this is also the approach with the lowest number of trainable parameters.

CVAE posterior collapse. While the CVAE baseline achieves competitive overall metrics, it exhibits posterior collapse: the posterior mean norm is near zero ($\|\mu_z\| \approx 0.03$) and the KL divergence is negligible ($\mathcal{L}_{\text{KL}} \approx 0.03$), indicating that the encoder has learned to match the prior $\mathcal{N}(0, I)$ rather than encode useful information. Because the FiLM layers are identity-initialized, a collapsed $z \approx 0$ produces $h' \approx h$, effectively recovering the base MTR model. The consequence is that the CVAE's latent space carries no interpretable structure: one cannot condition on a particular z to elicit a specific driving style, undermining the core purpose of a latent style variable. Our approach avoids this by design. The categorical latent directly selects which LoRA expert weights are active, creating a hard routing that the model cannot bypass. Each expert is forced to participate in prediction through structurally distinct parameter paths, rather than through an additive signal that can be tuned to identity. This structural integration also yields interpretable latents, as each discrete expert can be independently analyzed for kinematic and behavioral properties (Section IV-B).

In Table III, we stratify results by *Kalman difficulty* [36],

TABLE I: **Trajectory Prediction Benchmark Performance Comparisons.** We compare existing trajectory prediction models, TAE [27] (a behavior-aware trajectory predictor), and a CVAE baseline against our best-performing variant (Ours+MoV). Bolded values indicate best performance in respective columns.

Method	# Trainable Params	brierFDE↓	minADE↓	minFDE↓	MissRate↓
Autobot [28]	1.5M	2.4439	0.8892	1.7817	0.2803
Wayformer [29]	15.2M	2.5747	0.9348	1.9718	0.3574
MTR [30]	65.2M	2.1702	0.8645	1.7094	0.3107
MTR+Actions [26]	65.2M	2.1605	0.8658	1.7102	0.3208
TAE [27]	5.7M	2.9430±.070	1.0421±.040	2.2721±.069	0.4014±.017
CVAE [31]	**99.5K**	2.1603±.003	**0.8617**±.001	1.7032±.003	0.3164±.000
Ours+MoV [15]	195K	**2.1588**±.001	0.8619±.001	**1.7016**±.001	**0.3158**±.000

TABLE II: **Comparison of MoL Mixing Strategies.** We benchmark different parameter-efficient MoL schemes applied in our framework. All experiments are averaged over three fixed random seed runs. Bolded values indicate best-performance in respective columns. Our method improves upon the baseline model in overall trajectory prediction metrics across all MoL schemes. *The best all-around results are achieved with the MoV approach* [15], which uses IA3 [33] for finetuning, instead of LoRA.

Method	# Trainable Params	∇Router	brierFDE↓	minADE↓	minFDE↓	MissRate↓
MTR+Actions [26]	65.2M	-	2.1605	0.8658	1.7102	0.3208
Ours+Polytropon [16]	527K	✗	2.1602±.000	0.8623±.000	1.7037±.000	0.3164±.001
Ours+C-Poly [17]	969K	✗	2.1613±.002	0.8625±.001	1.7048±.002	0.3169±.000
Ours+HyperFormer [18]	527K	✗	2.1601±.001	0.8623±.000	1.7038±.001	0.3159±.001
Ours+CAT [34]	526K	✓	2.1607±.002	0.8624±.001	1.7041±.002	0.3171±.001
Ours+MoV [15]	195K	✓	**2.1588**±.001	**0.8619**±.001	**1.7016**±.001	**0.3158**±.000
Best % Improvement			0.066%	0.423%	0.497%	1.645%

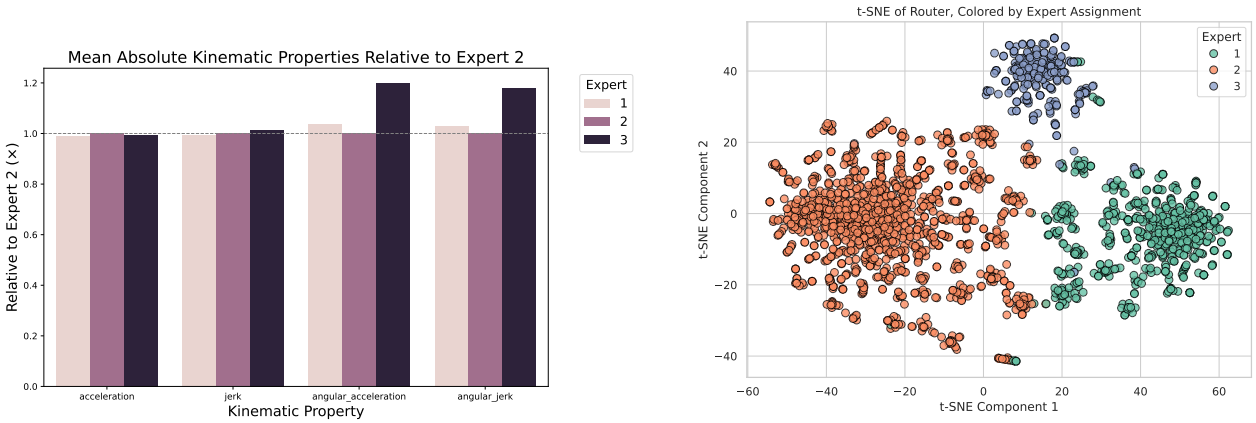
TABLE III: **minADE Comparison by Kalman Difficulty and TDBM Driving Style groups.** We compare existing trajectory prediction models, TAE, and our best-performing variant across Kalman difficulty categories and TDBM driving styles [35]. Bolded values indicate best-performance in respective columns. A per-variant breakdown of MoL schemes is provided in Table V of the Appendix.

	Kalman Difficulty			TDBM Driving Styles			
Method	Easy	Medium	Hard	Timid	Careful	Reckless	Threatening
Autobot [28]	0.8399	1.2436	1.9054	0.9186	0.9765	0.8906	0.8434
Wayformer [29]	0.8764	1.3440	3.0969	0.9676	0.9484	0.9362	0.8861
MTR [30]	0.8195	1.1736	3.0148	0.8849	0.7718	0.8670	0.8190
MTR+Actions [26]	0.8212	1.1781	2.5331	0.8846	0.7904	0.8687	0.8194
TAE [27]	0.9613	1.6102	3.8608	1.0738	0.9285	0.9742	1.0457
CVAE [31]	**0.8176**	1.1703	**2.5498**	**0.8795**	**0.8382**	**0.8162**	0.8646
Ours+MoV [15]	0.8180	**1.1690**	2.5502	0.8803	0.8652	0.8649	**0.8150**

[25] and *Trajectory to Driver Behavior Mapping (TDBM) driving styles* [35]. Kalman difficulty is defined as the FDE between the ground-truth trajectory and the prediction of a linear Kalman filter; scenarios where the future deviates substantially from constant-velocity extrapolation are classified as harder. TDBM labels are computed from trajectory-derived features (longitudinal jerk, lane-following deviation, and relative speed to neighbors) via a regression model calibrated through a human user study, producing four behavior categories: timid, careful, reckless, and threatening. Our method improves on medium-difficulty and threatening-style scenarios, while the CVAE baseline shows stronger performance on easy scenarios. A per-variant breakdown across MoL schemes is provided in Table V of the Appendix.

B. Interpreting Expert Specializations

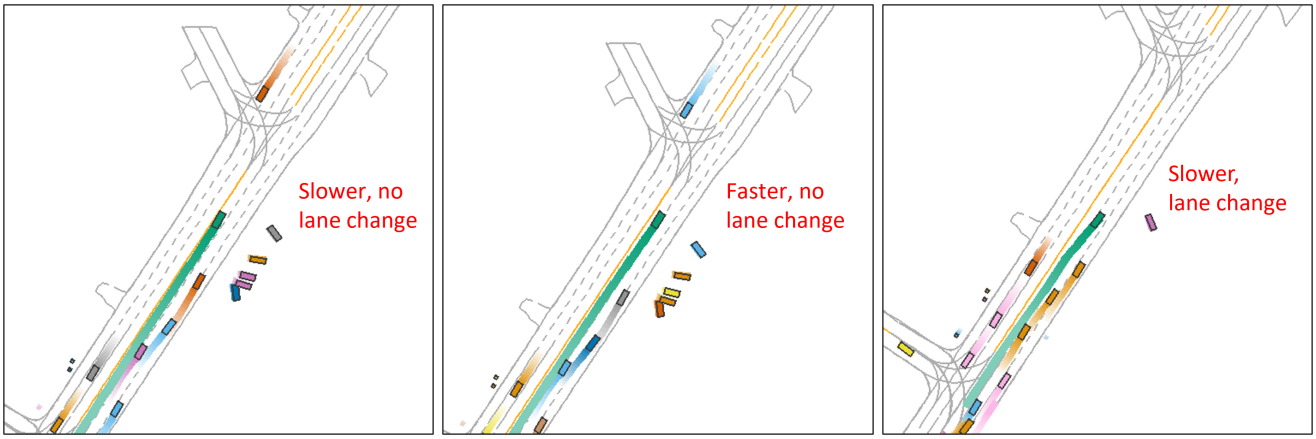
To investigate what behavior each expert specializes in, we compare second-order kinematic properties of predicted trajectories by expert assignment. We plot relative magnitudes of acceleration, jerk, angular acceleration, and angular jerk in Figure 3a, finding that experts differ in kinematic magnitudes consistent with distinct driving style profiles. t-SNE projections of router embeddings (Figure 3b) confirm that the router learns a clear three-class separation among social forces features; reconstruction loss converges well, implying that three latent classes are sufficient. We also show qualitative closed-loop rollouts in Figure 4, where different experts produce distinct trajectories in the same scenario. These rollouts are generated by iteratively re-planning with an MPC controller that tracks the prediction of



(a) Relative magnitudes of second-order kinematics.

(b) t-SNE visualization of router embeddings.

Fig. 3: **Visualizations of expert representations in Ours+CAT.** In (a), we show the mean magnitudes of acceleration, jerk, angular acceleration, and angular jerk between experts, relative to Expert 2. Expert separability is more pronounced in lateral (angular) kinematics than in longitudinal kinematics (acceleration, jerk), suggesting that *differences between experts are better captured by steering behavior than by speed profiles alone*. In (b), we project router embeddings to 2D with t-SNE and color each point by the router's expert assignment. The clear three-cluster separation confirms that our discrete routing does not suffer from expert collapse, and that the router learns meaningful groupings in the social forces feature space.



(a) Expert 1 (prior=0.3)

(b) Expert 2 (prior=0.6)

(c) Expert 3 (prior=0.1)

Fig. 4: **Qualitative result: Rollouts on a lane change scenario with different experts forced at test time.** Each panel shows the same Argoverse 2 scenario rolled out in closed loop: an MPC controller iteratively tracks the predictions of a single expert forced at test time (the underlying predictor itself is trained open-loop; see Section V). In each panel, the ego vehicle's rolled-out trajectory is drawn as the green path, surrounding traffic agents are drawn as colored boxes with their trajectories fading backward in time, and the red annotation summarizes the resulting expert behavior. Experts 1 and 2 both remain in the current lane, with Expert 2 traveling at a higher velocity than Expert 1. Expert 3, which has the lowest prior probability (i.e., the rarest driving style under the NHTSA-inspired prior), produces a lower-velocity rollout that changes lanes.

one forced expert; note that the predictor itself is trained and quantitatively evaluated open-loop. The lowest-prior expert predicts trajectories with lower velocity and a lane change, while the highest-prior expert maintains the current lane with higher speed. Visualizations for all MoL variants are provided in the Appendix.

C. Ablation Study

We conduct an ablation study on router expressivity, full fine-tuning, and the contribution of each loss term (Table IV). *Positive values indicate degradation relative to the full model, and negative values indicate improvement.* The greatest degradation in performance occurs when the reconstruction process or KL divergence is removed; without the KL loss, the router experiences expert collapse, as there is no pressure to maintain separation between the latent classes. Removing the

TABLE IV: **Ablation Study.** We evaluate the impact of each component in our MoL training scheme using the Ours+CAT variant. Positive values indicate degradation relative to the full model. We observe that removing the reconstruction loss causes the largest degradation in brierFDE and minFDE, while removing the KL loss causes the worst Miss Rate due to expert collapse.

Ours	brierFDE	minADE	minFDE	MissRate
+ LinearRouter	0.153%	-0.024%	0.226%	-0.157%
+ Full Finetuning	0.114%	-0.444%	0.313%	0.673%
- Social Forces	0.221%	-0.299%	0.401%	1.060%
- Context Features	0.186%	-0.096%	0.363%	1.199%
- KL Loss	0.243%	-0.136%	0.439%	1.669%
- Reconstruction	0.500%	0.071%	0.780%	1.337%
- Expert Entropy Loss	0.099%	0.096%	0.135%	-0.265%

reconstruction loss has the largest single impact on brierFDE and minFDE, confirming that the social forces reconstruction signal is critical for learning meaningful expert assignments. Removing social forces features or context features from the router input also degrades Miss Rate substantially, indicating that both feature sources contribute to routing quality. Full fine-tuning ($\sim$65M parameters) performs comparably to parameter-efficient learning, validating the MoL approach as an efficient alternative. The Appendix also includes a sweep over the number of latent classes, full per-seed standard deviations, and LoRA rank experiments.

D. Discussion: Trade-offs and Scope of Claims

Complexity versus performance gain. Our framework adds a router, expert adapters, and three auxiliary loss terms on top of the frozen backbone, while the resulting improvements on aggregate displacement metrics are admittedly small (sub-percent over MTR+Actions in Table II). We argue the added architecture is justified on grounds other than raw accuracy. First, the overhead is modest: the best-performing variant (Ours+MoV) trains only 195K parameters, roughly 0.3% of the 65.2M-parameter backbone, and the backbone itself is untouched, so base performance is preserved by construction. Second, the primary value of the framework is *controllability and interpretability* rather than benchmark gains: the discrete experts constitute a swappable, portable "style knob" that can be forced at test time for counterfactual rollouts (Figure 4) and analyzed independently for kinematic structure (Section IV-B) — capabilities the base model does not offer at any parameter cost. Under this framing, matching the backbone's accuracy while adding a controllable latent is the intended outcome.

Style versus other behavioral factors. We also wish to be precise about the scope of our "driving style" claim. Our analysis shows that the learned experts are kinematically distinct (Figure 3a), separable in the router's social-forces feature space (Figure 3b), and behaviorally distinct under forced rollouts. These properties are consistent with driving style, but they do not uniquely identify it: without standardized style annotations, we cannot fully disentangle style from correlated factors such as behavior mode, scene difficulty, or local interaction patterns, as noted in our limitations. Quantifying the mapping between learned experts and standardized style labels (e.g., MDSI [37] or TDBM [35] categories) is an important direction for future work.

Strength of the CVAE baseline. Finally, we note that the posterior collapse observed in our CVAE baseline could potentially be mitigated with anti-collapse training strategies (e.g., KL annealing or free bits), different latent dimensionalities, or alternative discrete-latent designs; our comparison should therefore be read as evidence that the discrete expert structure avoids collapse *by construction* — the categorical latent selects parameter paths that cannot be tuned to identity — rather than as a claim that no continuous-latent design could compete.

V. Conclusion and Limitations

In this paper, we introduced a framework based on Mixture-of-LoRA (MoL) to extract driving style variables from pre-trained trajectory models. The framework is both parameter-efficient and modular, making it easy to adapt to existing trajectory prediction works. This framework allows for controllable routing due to the global router, which learns a latent variable for driving style. Across other MoL approaches, *our latent driving style framework improves performance in nearly every case, especially for difficult scenarios and edge-case agent behavior*, with Mixture-of-Vectors (MoV) being the best-performing and most efficient paradigm to use with our approach.

Limitations. Firstly, we designed our framework based on one-shot, regressor-type trajectory models. Thus, predictions are not closed-loop. (The qualitative rollouts in Figure 4 close the loop externally with an MPC tracking controller at test time; training and all quantitative evaluation remain open-loop.) Secondly, a close-call trajectory prediction benchmark does not exist at the time of this work. In other words, our analysis on edge cases depends on the presence of rare samples in large-scale datasets, specifically Argoverse 2 in experiments. Thirdly, we assume that driving style is correlated with agent kinematic properties and social forces; this was a simplified assumption based on previous work. It is possible that there are other confounding factors associated with the extracted latent variable, such as spurious correlations within the data.

Acknowledgements

Large language models (LLMs) were used as assistive tools during this work. Specifically, LLMs were used to aid with revisions on human-written manuscript drafts, to assist with the implementation of the CVAE baseline method, and to

help develop plotting and visualization utilities. All LLM-generated content was reviewed and validated by the authors. This project is supported in part by Dr. Barry Mersky and Capital One E-Nnovate Endowed Professorships and UMD-ARL Cooperate Agreement.

REFERENCES

[1] S. Kim, S. J. Guy, D. Manocha, and M. C. Lin, "Interactive simulation of dynamic crowd behaviors using general adaptation syndrome theory," in *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, 2012, p. 55–62.

[2] E. J. Hu, yelong shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-rank adaptation of large language models," in *International Conference on Learning Representations*, 2022.

[3] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Physical review E*, vol. 51, no. 5, p. 4282, 1995.

[4] G. Ferrer, A. Garrell, and A. Sanfeliu, "Robot companion: A social-force based approach with human awareness-navigation in crowded environments," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2013, pp. 1688–1694.

[5] R. Karagulle, N. Aréchiga, A. Best, J. DeCastro, and N. Ozay, "A safe preference learning approach for personalization with applications to autonomous vehicles," *IEEE Robotics and Automation Letters*, vol. 9, no. 5, pp. 4226–4233, 2024.

[6] M. L. Schrum, E. Sumner, M. C. Gombolay, and A. Best, "Maveric: A data-driven approach to personalized autonomous driving," *IEEE Transactions on Robotics*, vol. 40, pp. 1952–1965, 2024.

[7] X. Liu, Y. Wang, Z. Zhou, K. Nam, C. Wei, and C. Yin, "Trajectory prediction of preceding target vehicles based on lane crossing and final points generation model considering driving styles," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 9, pp. 8720–8730, 2021.

[8] J. Hao, H. Xie, F. Guo, Y. Chen, and K. Song, "Vehicle trajectory prediction with driving style identification and intention fusion," in *2024 8th CAA International Conference on Vehicular Control and Intelligence (CVCI)*, 2024, pp. 1–6.

[9] Y. Xing, C. Lv, and D. Cao, "Personalized vehicle trajectory prediction based on joint time-series modeling for connected vehicles," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 1341–1352, 2020.

[10] X. Wang, K. Tang, X. Dai, J. Xu, J. Xi, R. Ai, Y. Wang, W. Gu, and C. Sun, "Safety-balanced driving-style aware trajectory planning in intersection scenarios with uncertain environment," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 4, pp. 2888–2898, 2023.

[11] R. Chandra, A. Bera, and D. Manocha, "Stylepredict: Machine theory of mind for human driver behavior from trajectories," *CoRR*, vol. abs/2011.04816, 2020.

[12] D. Kim, H. Shon, N. Kweon, S. Choi, C. Yang, and K. Huh, "Driving style-based conditional variational autoencoder for prediction of ego vehicle trajectory," *IEEE Access*, vol. 9, pp. 169 348–169 356, 2021.

[13] Y. Gu, X. Wang, J. Z. Wu, Y. Shi, Y. Chen, Z. Fan, W. XIAO, R. Zhao, S. Chang, W. Wu, Y. Ge, Y. Shan, and M. Z. Shou, "Mix-of-show: Decentralized low-rank adaptation for multi-concept customization of diffusion models," in *Advances in Neural Information Processing Systems*, vol. 36, 2023, pp. 15 890–15 902.

[14] S. Dou, E. Zhou, Y. Liu, S. Gao, J. Zhao, W. Shen, Y. Zhou, Z. Xi, X. Wang, X. Fan *et al.*, "Loramoe: Revolutionizing mixture of experts for maintaining world knowledge in language model alignment," *arXiv preprint arXiv:2312.09979*, vol. 4, no. 7, 2023.

[15] T. Zadouri, A. Üstün, A. Ahmadian, B. Ermis, A. Locatelli, and S. Hooker, "Pushing mixture of experts to the limit: Extremely parameter efficient moe for instruction tuning," in *The Twelfth International Conference on Learning Representations*, 2024.

[16] E. M. Ponti, A. Sordoni, Y. Bengio, and S. Reddy, "Combining parameter-efficient modules for task-level generalisation," in *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, 2023, pp. 687–702.

[17] H. Wang, T. Sun, C. Jin, Y. Wang, Y. Fan, Y. Xu, Y. Du, and C. Fan, "Customizable combination of parameter-efficient modules for multi-task learning," in *The Twelfth International Conference on Learning Representations*, 2024.

[18] R. Karimi Mahabadi, S. Ruder, M. Dehghani, and J. Henderson, "Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks," in *Annual Meeting of the Association for Computational Linguistics*, 2021.

[19] J. Wang, K. Messaoud, Y. Liu, J. Gall, and A. Alahi, "Forecast-peft: Parameter-efficient fine-tuning for pre-trained motion forecasting models," *arXiv preprint arXiv:2407.19564*, 2024.

[20] R. Yuan, M. Abdel-Aty, Q. Xiang, Z. Wang, and X. Gu, "A temporal multi-gate mixture-of-experts approach for vehicle trajectory and driving intention prediction," *IEEE Transactions on Intelligent Vehicles*, vol. 9, no. 1, pp. 1204–1216, 2024.

[21] P. Xu, J.-B. Hayet, and I. Karamouzas, "Socialvae: Human trajectory prediction using timewise latents," in *European Conference on Computer Vision*. Springer, 2022, pp. 511–528.

[22] A. Van Den Oord, O. Vinyals *et al.*, "Neural discrete representation learning," *Advances in neural information processing systems*, vol. 30, 2017.

[23] S. G. Klauer, T. A. Dingus, V. L. Neale, J. D. Sudweeks, and D. J. Ramsey, "Comparing real-world behaviors of drivers with high versus low rates of crashes and near crashes," United States. National Highway Traffic Safety Administration, Tech. Rep. DOT HS 811 091, 2009.

[24] B. Wilson, W. Qi, T. Agarwal, J. Lambert, J. Singh, S. Khandelwal, B. Pan, R. Kumar, A. Hartnett, J. K. Pontes, D. Ramanan, P. Carr, and J. Hays, "Argoverse 2: Next generation datasets for self-driving perception and forecasting," in *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS Datasets and Benchmarks 2021)*, 2021.

[25] L. Feng, M. Bahari, K. M. B. Amor, É. Zablocki, M. Cord, and A. Alahi, "Unitraj: A unified framework for scalable vehicle trajectory prediction," in *European Conference on Computer Vision*. Springer, 2024, pp. 106–123.

[26] L. Zheng, S. Son, J. Liang, X. Wang, B. Clipp, and M. C. Lin, "Deep stochastic kinematic models for probabilistic motion forecasting in traffic," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 943–950.

[27] R. Jiao, X. Liu, B. Zheng, D. Liang, and Q. Zhu, "Tae: A semi-supervised controllable behavior-aware trajectory generator and predictor," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 12 534–12 541.

[28] R. Girgis, F. Golemo, F. Codevilla, M. Weiss, J. A. D'Souza, S. E. Kahou, F. Heide, and C. Pal, "Latent variable sequential set transformers for joint multi-agent motion prediction," in *International Conference on Learning Representations*, 2022.

[29] N. Nayakanti, R. Al-Rfou, A. Zhou, K. Goel, K. S. Refaat, and B. Sapp, "Wayformer: Motion forecasting via simple & efficient attention networks," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 2980–2987.

[30] S. Shi, L. Jiang, D. Dai, and B. Schiele, "Motion transformer with global intention localization and local movement refinement," *Advances in Neural Information Processing Systems*, vol. 35, pp. 6531–6543, 2022.

[31] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *Advances in Neural Information Processing Systems*, vol. 28, 2015.

[32] E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. C. Courville, "Film: Visual reasoning with a general conditioning layer," in *AAAI*, 2018.

[33] H. Liu, D. Tam, M. Mohammed, J. Mohta, T. Huang, M. Bansal, and C. Raffel, "Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning," in *Advances in Neural Information Processing Systems*, 2022.

[34] A. Prabhakar, Y. Li, K. Narasimhan, S. Kakade, E. Malach, and S. Jelassi, "Lora soups: Merging loras for practical skill composition tasks," 2024.

[35] E. Cheung, A. Bera, E. Kubin, K. Gray, and D. Manocha, "Identifying driver behaviors using trajectory features for vehicle navigation," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3445–3452.

[36] M. Bahari, S. Saadatnejad, A. Rahber, M. J. Shaverdikondori, A. H. Shahidzadeh, S.-M. Moosavi-Dezfooli, and A. Alahi, "Vehicle trajectory prediction works, but not everywhere," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 17 123–17 133.

[37] O. Taubman-Ben-Ari, M. Mikulincer, and O. Gillath, "The multidimensional driving style inventory—scale construct and validation," *Accident Analysis and Prevention*, vol. 36, no. 3, pp. 323–332, 2004.

APPENDIX

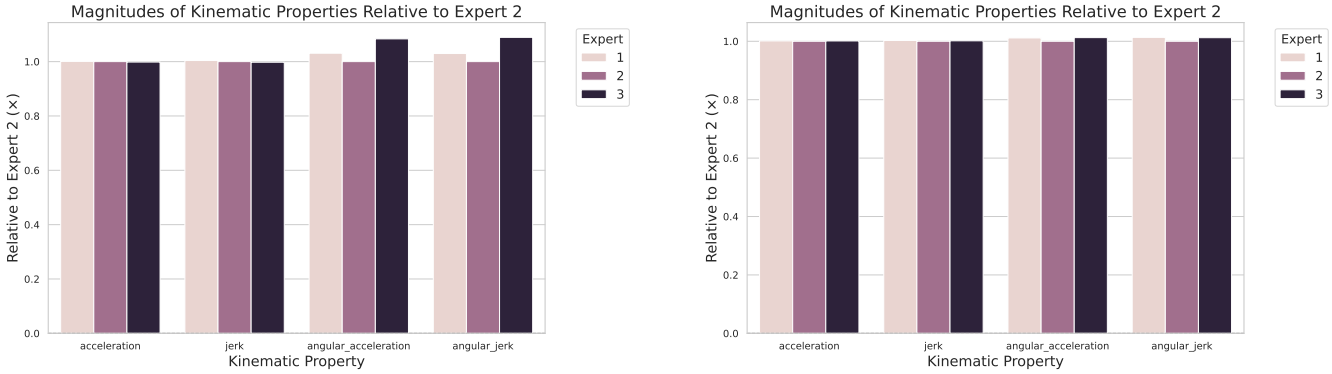
Our code can be found on the project website: laurayuzheng.github.io/polysona

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
```

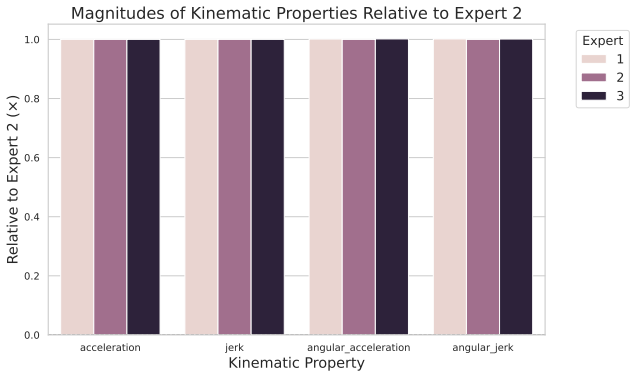
TABLE V: **minADE by Kalman Difficulty and TDBM Driving Style across MoL Mixing Strategies.** We compare variants of our method on different MoL schemes across Kalman difficulty categories and TDBM driving styles [35]. Bolded values indicate best performance in respective columns. In general, we find that *weight mixing provides the greatest improvement to samples with hard Kalman difficulty*. None of the variants improved minADE on "careful" driving samples; however, "careful" is also the TDBM driving style assigned to the vast majority of common driving scenarios.

	Kalman Difficulty			TDBM Driving Styles			
Method	Easy	Medium	Hard	Timid	Careful	Reckless	Threatening
MTR+Actions [26]	0.8212	1.1781	2.5331	0.8846	0.7904	0.8687	0.8194
Ours+PolyTropon [15]	0.8185	1.1690	2.5585	0.8803	0.8665	0.8653	0.8156
Ours+C-Poly [17]	0.8187	1.1691	2.5474	0.8803	0.8472	0.8655	0.8166
Ours+HyperFormer [18]	0.8186	**1.1673**	2.5707	0.8803	**0.8471**	0.8653	0.8156
Ours+CAT [34]	0.8188	1.1678	**2.4978**	**0.8793**	0.8477	0.8655	0.8161
Ours+MoV [15]	**0.8180**	1.1690	2.5502	0.8803	0.8652	**0.8649**	**0.8150**
Best % Improvement	0.360%	0.921%	1.404%	0.570%	-0.707%	0.397%	0.675%

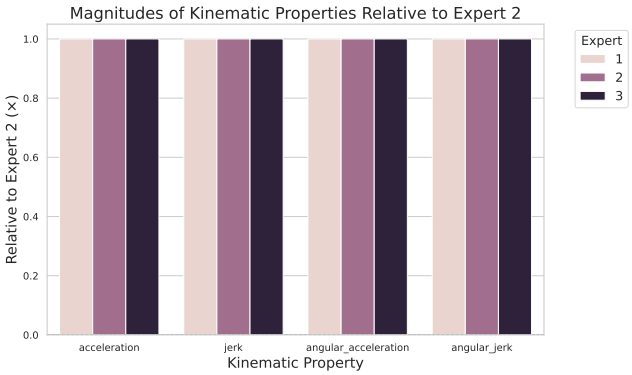


(a) C-Poly

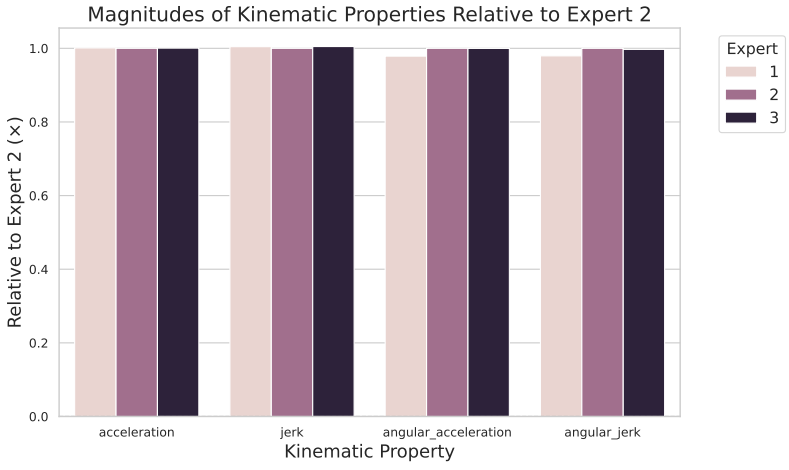
(b) HyperFormer



(c) Polytropon

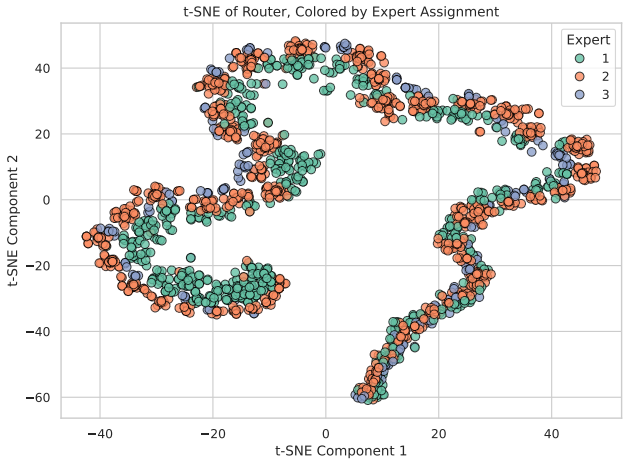


(d) MoV

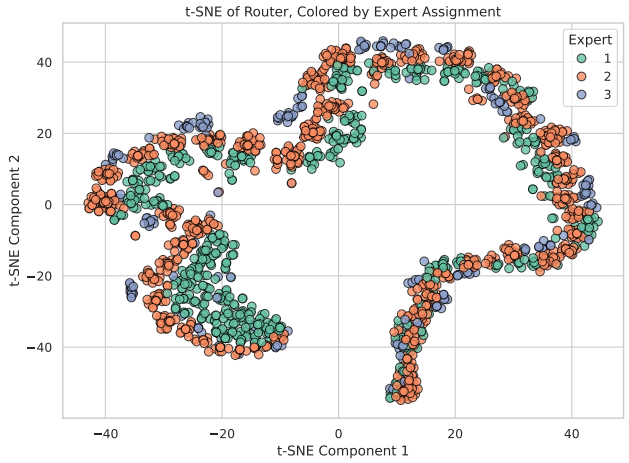


(e) CAT

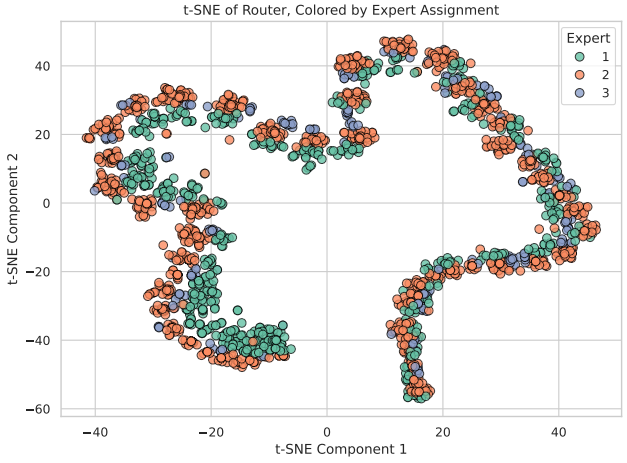
Fig. 5: **Kinematic magnitude comparisons for all variants of our approach.** Experiments run with seed 0 are plotted.



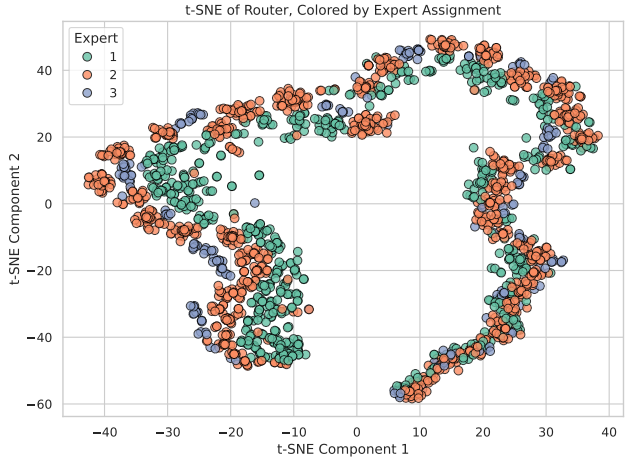
(a) C-Poly



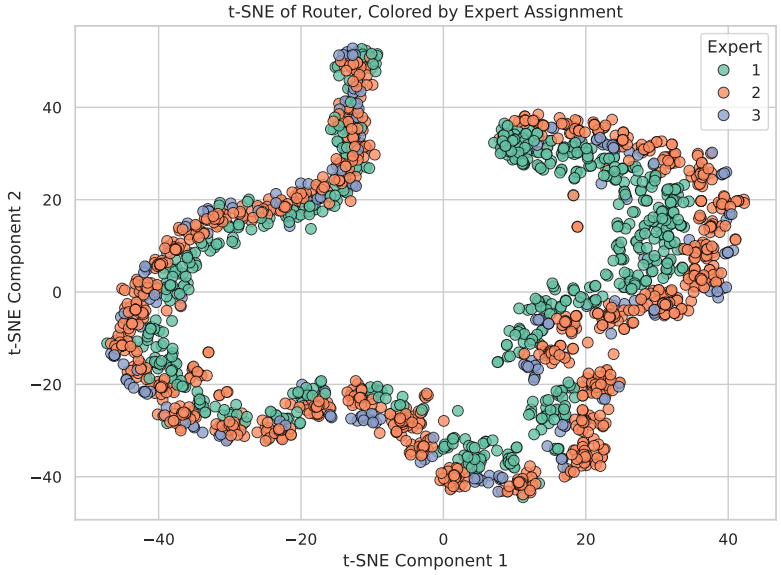
(b) HyperFormer



(c) Polytropon



(d) MoV



(e) CAT

Fig. 6: **t-SNE visualization of router embeddings for all variants of our approach.** Experiments run with seed 0 are plotted.

TABLE VI: 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

TABLE VII: 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

TABLE VIII: **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 IX: **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

TABLE X: **Standard Deviation of Trajectory Prediction Benchmark Performance Comparisons.**

Method	brierFDE↓	minADE↓	minFDE↓	MissRate↓
TAE [27]	0.0699	0.0395	0.0688	0.0165
CVAE [31]	0.0030	0.0006	0.0029	0.0003
Ours+Polytropon [16]	0.0003	0.0003	0.0003	0.0007
Ours+C-Poly [17]	0.0022	0.0009	0.0021	0.0002
Ours+HyperFormer [18]	0.0014	0.0003	0.0014	0.0008
Ours+CAT [34]	0.0015	0.0010	0.0015	0.0008
Ours+MoV [15]	0.0008	0.0006	0.0009	0.0003

TABLE XI: **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
TAE [27]	0.0347	0.0831	0.3868	0.0390	0.0411	0.0384	0.0397
Ours+PolyTropon [15]	0.0003	0.0032	0.0040	0.0005	0.0038	0.0004	0.0005
Ours+C-Poly [17]	0.0013	0.0003	0.0146	0.0009	0.0286	0.0012	0.0011
Ours+HyperFormer [18]	0.0006	0.0013	0.0060	0.0006	0.0330	0.0004	0.0007
Ours+CAT [34]	0.0012	0.0051	0.0682	0.0014	0.0340	0.0012	0.0012
Ours+MoV [15]	0.0007	0.0006	0.0041	0.0008	0.0026	0.0007	0.0006

TABLE XII: 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