Motion Style Transfer: Modular Low-Rank Adaptation for Deep Motion Forecasting (Supplementary Material)

Parth Kothari **Danya Li** **Yuejiang Liu** **Alexandre Alahi**
École Polytechnique Fédérale de Lausanne (EPFL)

A Additional Experiments

Training a residual has been shown effective in other domains like robotics [1]. Yet, it's not clear what kind of residual is most effective for adaptation, and where they should be introduced. To address these open questions, we introduced two key components specific to motion forecasting: (i) we model the residual as a low-dimensional bottleneck, (ii) we propose a modular adaptation strategy that facilitates a targeted choice of adaptation layers.

A.1 Importance of Low-Rank constraint

Tab. 1 illustrates the importance of the low-rank constraint in the residual design. Further, Tab. 2 provides the performance on varying the rank r of the MoSA modules on SDD. Very low rank limits the number of style factors that can be updated leading to sub-optimal performance. On the other hand, a high rank increases the number of trainable parameters, leading to overfitting in the low-data regime.

Sec. 3.3 demonstrated the effectiveness of our modularized adaptation strategy. Next, we provide additional experiments that further validates our proposed scheme.

A.2 Modularized Adaptation across Agent Motion on SDD

We demonstrate the efficacy of modularized adaptation scheme on the SDD dataset. We utilize the cyclists data on *deathCircle_0* scene. The training and adaptation data are constructed based on the average speed of the cyclist trajectories. The training set contains low-speed trajectories with the speed in the range of 0.5 to 3.5 pixel per second. The adaptation set has cyclist trajectories with an average speed in the range of 4 to 8 pixels per second. We use $N_{target} = \{20\}$ trajectories for adaptation. Given our setup, the dominant underlying style factor that changes across domains is the agent speed distribution (the scene context and the type of agent are fixed). We benchmark the performance of Y-Net-Mod given the five modularization strategies described in the main text. Rank of MoSA is set to 2.

Fig. 1 illustrates the performance of various modular adaptation strategies. The generalization error is very high as the error accumulates quickly for fast-moving agents. Using just 20 samples, updating only the agent motion module [A] leads to the best performance, while including the scene context module [S] during adaptation worsens performance.

A.3 Modularized Adaptation across Scenes on inD

In the main text, we demonstrated the efficacy of modularized adaptation strategy by training on pedestrian data from $\{scene2, scene3, scene4\}$ and testing on unseen scene $scene1$. Different scene combinations can help to corroborate the efficacy of our modular adaptation strategy. Therefore, as shown in Fig. 2, we provide the results of our strategy for a different setup of scene generalization. We train on $\{scene1, scene3, scene4\}$ and test on unseen scene $scene2$. We observe the same trend

6th Conference on Robot Learning (CoRL 2022), Auckland, New Zealand.

Table 1: Importance of the low-rank constraint.

	Lyft Level 5		
N_{target} (batches)	8	15	36
Finetuning	41.76 ± 0.45	33.90 ± 0.89	31.83 ± 1.89
No-Rank Constraint	42.37 ± 0.72	34.12 ± 0.51	30.37 ± 2.19
Rank Constraint	$\mathbf{29.43 \pm 0.70}$	$\mathbf{25.96 \pm 0.47}$	$\mathbf{25.06 \pm 1.26}$

Table 2: Varying the rank r of MoSA.

	Stanford Drone		
N_{target}	10	20	30
Rank 1	51.84 ± 1.41	46.68 ± 1.32	42.53 ± 1.19
Rank 3	$\mathbf{49.98 \pm 1.05}$	$\mathbf{45.55 \pm 0.77}$	$\mathbf{41.69 \pm 0.88}$
Rank 10	51.44 ± 0.66	46.48 ± 0.84	42.44 ± 0.72

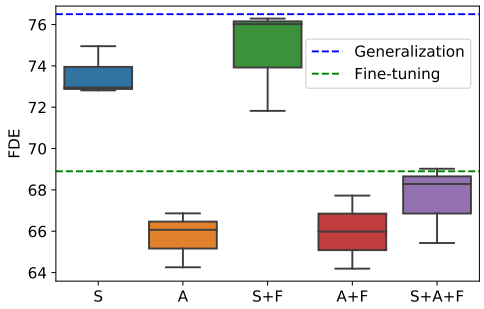


Figure 1: Agent Motion Transfer from *slow* cyclists to *fast* cyclists on *death_circle* SDD scene with $N_{target} = 20$ samples using different MoSA configurations. [A] performs best while [S] worsens performance.

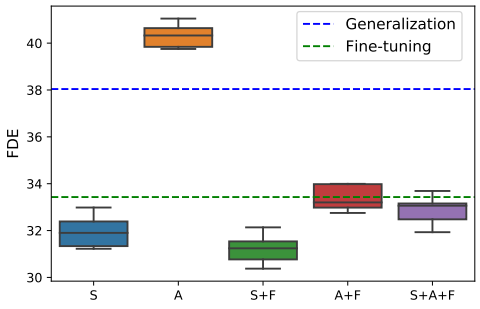


Figure 2: Scene Transfer from *scene1, scene3, scene4* to *scene2* on InD *pedestrians* with $N_{target} = 20$ samples using different MoSA configurations. [S+F] performs best while [A] worsens performance.

as Fig. 9 of main text: updating the scene module [S] is more effective than updating all the modules [S+A+F], and adapting of agent motion module [A] deteriorates performance.

B Additional visualizations

Motion Style Transfer across Scenes on inD. In this experiment, the Y-Net model is trained on pedestrians in $\{scene2, scene3, scene4\}$ and tested on unseen scene $scene1$. We qualitatively show the goal decoder output difference of three examples in Fig. 3. We can observe that the adapted model learns to cross at a particular location, even though this behavior is not present in the training data.

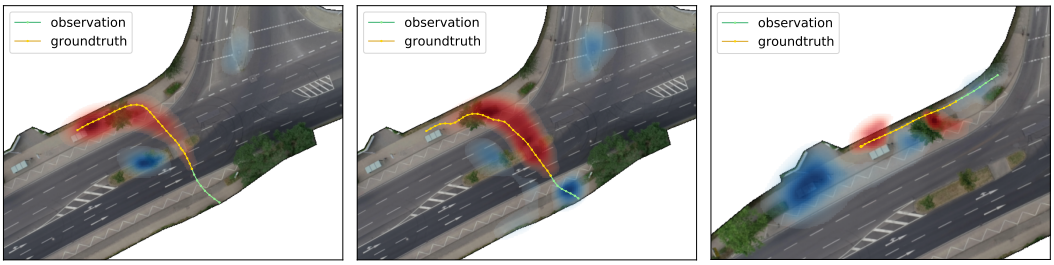
Motion Style Transfer across Scenes on Level 5. In this experiment, we divide the L5 dataset into two splits based on the data collection locations thereby, constructing a scene style shift scenario (see Fig.7). In addition to the examples provided in the main text, Fig. 4 qualitatively illustrates the improvement of the attention heatmaps of the last layer of ViT post model adaptation using MoSA. MoSA helps to better focus on the different possible future routes and the vehicles in front.

C Dataset Preparation

We use a total of three datasets to study the performance of motion style adapters: Stanford Drone Dataset (SDD) [2], the Intersection Drone Dataset (InD) [3], and Level 5 Dataset (L5) [4].

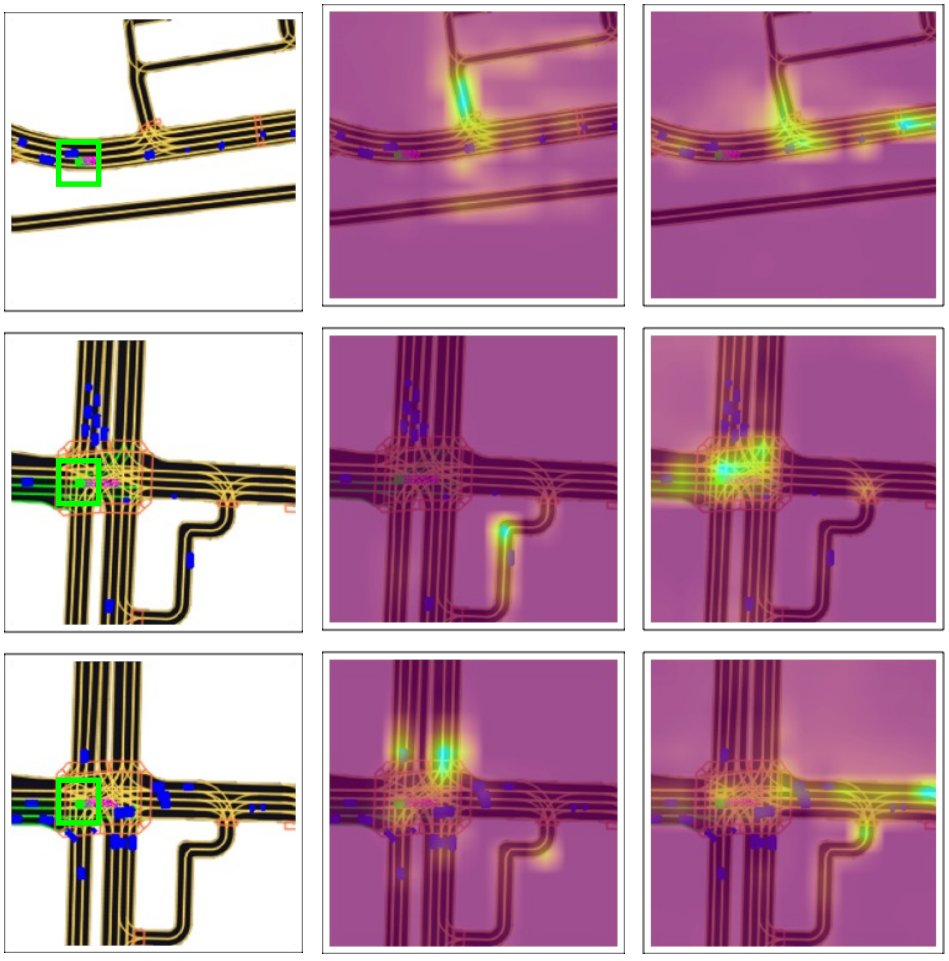
C.1 Stanford Drone Dataset (SDD)

SDD comprises 20 top-down scenes on the Stanford campus with various agent types (i.e., pedestrians, bicyclists, car, skateboarders, buses, golf carts). We perform short-term prediction where we give 3.2 seconds trajectories and output the future 4.8 seconds. Following the same pre-processing procedure in Mangalam et al. [5], we filter out short trajectories below 8 seconds in duration, split temporally discontinued trajectories, and then use a sliding window approach without overlap to split the cleaned trajectories. After those steps, the dataset contains 14860 pedestrian trajectories and 5152 bicyclist trajectories. The semantic segmentation has 6 classes, namely pavement, terrain, structure, tree, road, and others [6].



(a) Sample 1 (b) Sample 2 (c) Sample 3

Figure 3: Illustration of the difference in goal decoder output of Y-Net on the scene style transfer on InD using our motion style adapters (Red is positive, blue is negative). Observe that in the adapted scene, pedestrians tend to cross at a particular segment of the road. This behavior did not occur during the pre-training. MoSA learns this novel behavior using just 30 samples of adaptation.



(a) Input sample (b) Pre-adaptation (c) Post-adaptation

Figure 4: Attention heatmaps of the last layer of ViT before and after model adaptation on unseen route in Level 5. Post adaptation, the attention maps are more refined. The ego (in green box) better focuses on the different possible future routes and the vehicles in front.

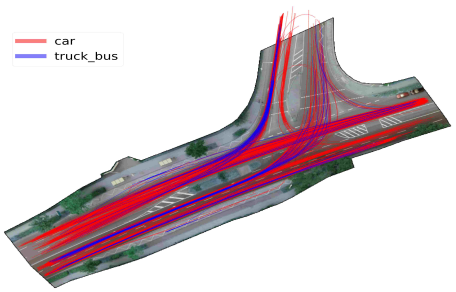


Figure 5: Car and truck trajectory distribution in inD *scene1*. The two agents share the same scene context, differing in the motion style.

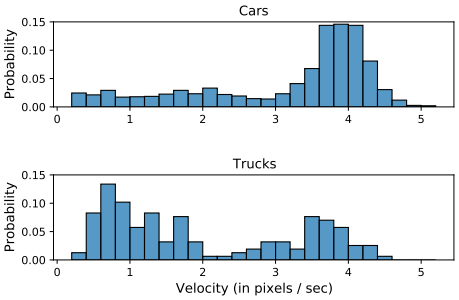


Figure 6: Velocity distribution of *car* and *truck* in inD *scene1*. Static samples removed.

C.2 Intersection Drone Dataset (inD)

The inD Dataset comprises four distinct road intersections, namely *scene1*, *scene2*, *scene3*, and *scene4*, with various agent types, i.e., cars, pedestrians, bicyclists, trucks and buses. We perform both short-term and long-term predictions. The short-term prediction outputs 4.8 seconds trajectories given 3.2 seconds observation, while the long-term prediction generates 30 seconds future trajectories given 5 seconds of observed ones. We use similar pre-processing steps as for SDD. Additionally, we convert the data from the real-world coordinates to pixel coordinates using the provided scaling factors [3]. After the pre-processing steps, inD contains 1396 long-term pedestrian trajectories, 1508 short-term car trajectories, and 157 short-term trucks trajectories.

C.3 Lyft Level 5 Dataset (L5)

Lyft Level 5 Prediction is a self-driving dataset, containing over 1,000 hours of real-world vehicle data, where each scene lasts 25 seconds. We perform long-term motion forecasting where the ego vehicle moves in a closed loop for the entirety of the scene for 25 seconds, while the surrounding agents follow log-replay [4]. In the real-world, ego vehicles can come across novel scene contexts, for example, road constructions.

In summary, SDD and inD with different heterogeneous agents provide the ideal setup to validate motion style transfer techniques. L5 dataset provides long sequences of ego vehicle to study the effects of style transfer on long-term self-driving settings.

D Pre-trained Models

We utilize the state-of-the-art model Y-Net [5] for experiments on SDD and inD. We further propose an alternate design of Y-Net termed Y-Net-Mod to demonstrate the efficacy of our modular adaptation strategy. Finally, we utilize the ViT-Tiny [7] architecture for experiments on L5.

D.1 Y-Net

Y-net [5] comprises three sub-networks: the scene heatmap encoder, the waypoint heatmap decoder, and the trajectory heatmap decoder. Specifically, the encoder is designed as a U-net encoder which consists of one center convolutional layer, four intermediate blocks where each uses max pooling and two convolutional layers, and one final max pooling layer. It takes as input the concatenation of the scene semantic map and past trajectory heatmap.

D.2 Y-Net-Mod

We construct Y-Net-Mod on top of the original Y-Net architecture. The modification treats the scene context and agent motion *independently* before fusing their representations together. As shown in Fig. 2 of main text, the first three layers of the original encoder are decoupled into scene context and past agent motion modules in order to learn their representations independently. Subsequently, the

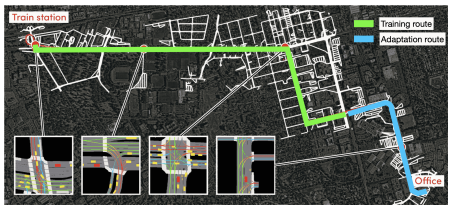


Figure 7: Train-adaptation split for Level 5 dataset [4]. Model is trained on the green route and adapted on the blue route.

Table 3: Generalization performance of Y-Net and Y-Net-Mod on the modularization setups of inD. Errors reported are Top-20 ADE/FDE in pixels. Y-Net-Mod does not degrade performance in comparison to Y-Net.

Experiment	Y-Net	Y-Net-Mod
Scene transfer	6.44 / 10.72	6.60 / 11.17
Agent motion transfer	24.48 / 33.54	24.56 / 29.07

representations are fused together using the fusion encoder, that is similar in design to the last two layers of Y-Net. The original number of channels in each encoder layer of Y-Net are evenly divided between each module in Y-Net-Mod encoder so that the latter is compatible with the Y-Net decoders.

Benchmarking Y-Net and Y-Net-Mod: To illustrate that our modification to Y-Net does not result in severe drop in performance, we benchmark the performance of Y-Net and Y-Net-Mod on inD for the modularization setups presented in main draft: 1) scene transfer of pedestrians, 2) agent motion transfer from cars to trucks. Table. 3 illustrates that modularization of the Y-Net encoder does not lead to a significant drop in the performance.

D.3 Vision Transformer

We utilize the official ViT-Tiny architecture [7] for the Level 5 dataset. We only modify the last layer to output the forecasting predictions in the form of x, y coordinates for T_{pred} time-steps.

E Implementations Details

We present implementation details and hyperparameters for each method and model training. For each experiment, the best model is chosen based on the performance on the validation set. All model pretraining follows the training, validation and test split of 7:1:2. During adaptation, all experiments utilize Adam optimizer and a batch size of 10, unless mentioned otherwise. Learning rate for FT is 5e-5, 5e-4 for ET, 5e-5 for PA, 1e-4 for BN, and 5e-3 for MoSA, unless mentioned otherwise. The details for our designed experiments are listed as below.

Motion Style Transfer across Agents on SDD. We pretrain Y-Net network for 100 epochs and learning rate of 5e-5. Rest of the hyper-parameters are kept the same as [5]. We adapt the pretrained model using $N_{target} = \{10, 20, 30\}$ trajectories and utilize 80 trajectories for validation. We adapt the pretrained model for 100 epochs with an early stop value of 30 epochs. For MoSA, the rank value is set to 3.

Motion Style Transfer across Scenes on inD. In this experiment, we utilize the pretrained model provided by Y-Net paper [5]. We adapt this model using $N_{target} = \{20\}$ trajectories and use 40 trajectories for validation. The pretrained model is 100 epochs. Fig. 3 illustrates the adaptation performance of MoSA using 30 samples. MoSA learns the unseen behavior of pedestrians crossing at a particular segment of the road, that was unobserved in the training scenes of inD.

Motion Style Transfer across Scenes on L5. To simulate a scene context transfer scenario, we split the dataset as shown in Fig. 7. The ViT-Tiny model is trained on the majority route shown in green and adapted to the blue route that was not seen during training. We follow the training strategy provided in Houston et al. [4]: specifically, the network is trained to accept BEV rasters of size 224×224 pixels (centered around the SDV) to predict future (x, y) positions over a 1.2 second horizon. The hyper-parameters of the ViT-Tiny architecture are kept the same as in [7]. We train the model on the training data corresponding to the majority route for 15 epochs using batch size of 64.

To simulate low-resource settings during adaptation, the network is shown the unseen route only once, sampled at different rates. As a result, we adapt for $N_{batches} = \{4, 8, 15, 24, 36\}$ with a batch

size of 64. We benchmark the following four cases: 1) Full model fine-tuning with learning rate of 1e-4, 2) Final layer fine-tuning with learning rate of 3e-4, 3) Adaptive layer normalization with a learning rate of 1e-4, 4) MoSA (ours) with rank value of 8 and learning rate of 3e-3. We apply MoSA across the query and value matrices of each attention layer. We observe that applying MoSA across the feed-forward layers deteriorated performance. We adapt all the methods for 250 epochs using a one-cycle learning rate scheduler.

Motion Style Transfer across Agents on inD with Modular Adaptation. We perform style transfer from cars to trucks in *scene1* of inD. Cars and trucks data have different speed distribution (see Fig. 6) but share the same context as shown in Fig. 5. Trajectories with an average speed less than 0.2 pixels per second are filtered out. We train a Y-Net-Mod model on cars and adapt the model to trucks in which $N_{target} = \{20\}$ trajectories are used for adaptation and 40 trajectories for validation.

Motion Style Transfer across Scenes on inD with Modular Adaptation. We pretrained Y-Net-Mod model using pedestrian data from scene ids = $\{2, 3, 4\}$ and transferred to pedestrian data from *scene1* following the setup in [5]. The adaptation uses $N_{target} = \{20\}$ for fine-tuning and 20 trajectories for validation.

Motion Style Transfer across Agent Motion on SDD with Modular Adaptation. We pretrain a Y-Net-Mod model using slow cyclists from *deathCircle_0* scene and adapt to fast cyclists from the same scene. The pretraining set has 1213 trajectories. The adaptation set has 381 trajectories where 50 trajectories are used for validation.

E.1 Initialization of Motion Style Adapters

Matrices A and B are initialized such that the original network is not affected when training starts. Specifically, we use a random Gaussian initialization for A and zero for B. This initialization scheme allows these modules to be ignored at certain layers if there is no need for a change in activation distribution.

References

[1] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine. Residual reinforcement learning for robot control. *2019 International Conference on Robotics and Automation (ICRA)*, pages 6023–6029, 2019.

[2] A. Robicquet, A. Sadeghian, A. Alahi, and S. Savarese. Learning social etiquette: Human trajectory understanding in crowded scenes. In *ECCV*, 2016.

[3] J. Bock, R. Krajewski, T. Moers, S. Runde, L. Vater, and L. Eckstein. The ind dataset: A drone dataset of naturalistic road user trajectories at german intersections. *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 1929–1934, 2020.

[4] J. L. Houston, G. C. A. Zuidhof, L. Bergamini, Y. Ye, A. Jain, S. Omari, V. I. Iglovikov, and P. Ondruska. One thousand and one hours: Self-driving motion prediction dataset. In *CoRL*, 2020.

[5] K. Mangalam, Y. An, H. Girase, and J. Malik. From goals, waypoints & paths to long term human trajectory forecasting. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15213–15222, 2021.

[6] H. Caesar, J. Uijlings, and V. Ferrari. Coco-stuff: Thing and stuff classes in context. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1209–1218, 2018.

[7] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020.