Supplementary Materials: Hierarchical Latent Structure for Multi-Modal Vehicle Trajectory Forecasting

Dooseop Choi[1] and KyoungWook Min[1]

Artificial Intelligence Research Laboratory, ETRI
{d1024.choi,kwmin92}@etri.re.kr

A Visualization of Vehicle-Lane Interaction (VLI)

As we mentioned in the paper, for the calculation of the lane-level context vector $\mathbf{a}_i^m$, we use not only the reference lane but also the surrounding lanes with their relative importance. This idea is based on the fact that human drivers often pay attention to surrounding lanes when driving along the reference lane. To show how our model pays attention to the surrounding lanes for the target vehicle, we use four scenarios in nuScenes and show the results in Figure 1. In the figure, blue lines denote the reference lanes while the others denote the surrounding lanes. The surrounding lanes of high importance are shown in red and the surrounding lanes of low importance are shown in green. We can see in the figure that our forecasting model pays more attention to the surrounding lanes that are close to the reference lane.

B Mode Blur in SOTA Model

We show in Figure 2 the prediction examples of the state-of-the-art model [6]. We note here that the figure is identical to the figure illustrated in the supplementary material of [6]. The model is built upon [3], which is based on the VAE framework and learns a diverse joint distribution over multi-agent future trajectories in a traffic scene. In the figure, green and light blue bounding boxes respectively denote the AV and surrounding vehicles. The solid lines with light blue dots denote the predicted trajectories for the surrounding vehicles. We can see in the figure that some trajectories are located between adjacent lanes, which can cause uncomfortable rides for the AV with plenty of sudden brakes and steering changes [4].

C Further Explanation to Average Quality

We mentioned in the paper that ADE_1 and FDE_1 metrics shown in the tables presented in the paper represent the average quality of the trajectories generated

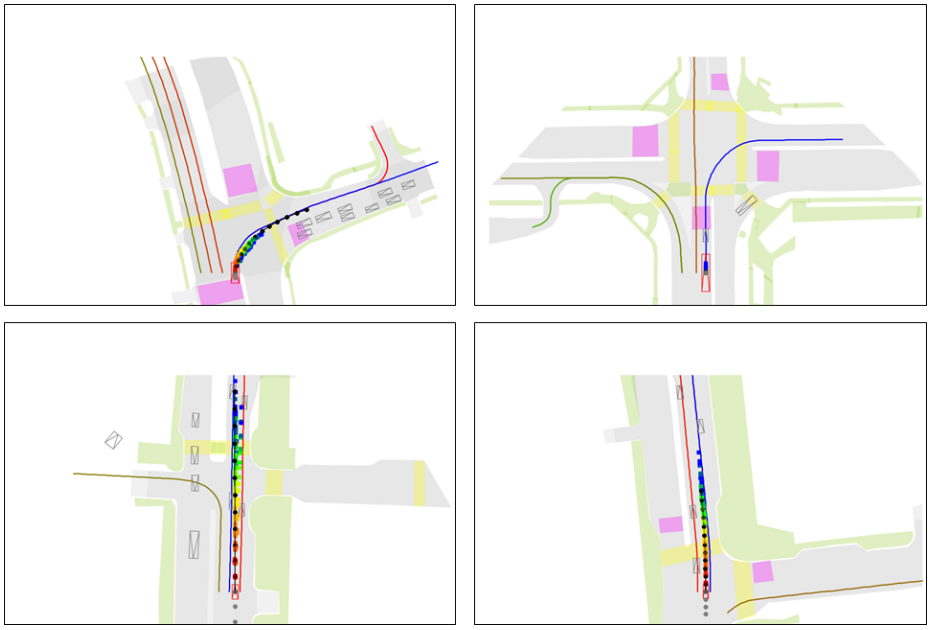

Fig. 1: VLI visualization

for the ground-truth trajectory $\mathbf{Y}$. The ADE_1 metric in the table is calculated as

$$ADE_1 = \frac{1}{|\mathcal{D}|} \sum_{\mathbf{Y} \in \mathcal{D}} ADE(\hat{\mathbf{Y}}, \mathbf{Y}), \tag{1}$$

where $\mathcal{D}$ is the test dataset and $\hat{\mathbf{Y}}$ is the prediction of $\mathbf{Y}$. Because there are relatively few distinct actions that can be taken by a vehicle over a reasonable time horizon (3 to 6 seconds) [11], the ground-truth trajectories in $\mathcal{D}$ can be clustered into multiple groups, where the trajectories of each group are very close to each other in Euclidean space. Assume that there are N groups in $\mathcal{D}$ and let $\mathcal{Y}_i$ denote the i-th group. Then Eqn. 1 can be expressed as

$$\begin{aligned} ADE_1 &= \frac{1}{|\mathcal{D}|}\{ \sum_{\mathbf{Y} \in \mathcal{Y}_1} ADE(\hat{\mathbf{Y}}, \mathbf{Y}) + ... + \sum_{\mathbf{Y} \in \mathcal{Y}_N} ADE(\hat{\mathbf{Y}}, \mathbf{Y})\} \\ &= \frac{|\mathcal{Y}_1|}{|\mathcal{D}|} \frac{1}{|\mathcal{Y}_1|} \sum_{\mathbf{Y} \in \mathcal{Y}_1} ADE(\hat{\mathbf{Y}}, \mathbf{Y}) + ... + \frac{|\mathcal{Y}_N|}{|\mathcal{D}|} \frac{1}{|\mathcal{Y}_N|} \sum_{\mathbf{Y} \in \mathcal{Y}_N} ADE(\hat{\mathbf{Y}}, \mathbf{Y}) \\ &= w_1 \frac{1}{|\mathcal{Y}_1|} \sum_{\mathbf{Y} \in \mathcal{Y}_1} ADE(\hat{\mathbf{Y}}, \mathbf{Y}) + ... + w_N \frac{1}{|\mathcal{Y}_N|} \sum_{\mathbf{Y} \in \mathcal{Y}_N} ADE(\hat{\mathbf{Y}}, \mathbf{Y}) \\ &= w_1 AADE(\mathcal{Y}_1) + ... + w_N AADE(\mathcal{Y}_N), \end{aligned} \tag{2}$$

where $\sum_{i=1}^{N} w_i = 1$. Since the trajectories of each group are very close to each other in Euclidean space, $AADE(\mathcal{Y}_i)$ in the last line of Eqn. 2 can be approxi-

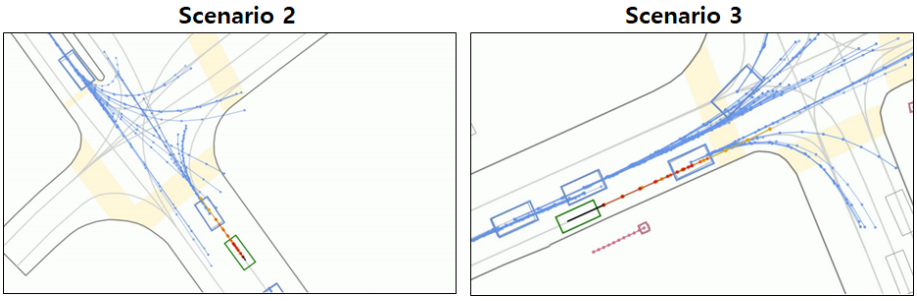

Fig. 2: Trajectory prediction examples of [6]

Model	ADE_1/FDE_1	ADE_{15}/FDE_{15}
Ours+Multi	2.64/6.32	0.89/1.72
Ours+Single	2.64/6.32	0.97/1.95

(a) nuScenes

Model	ADE_1/FDE_1	ADE_{12}/FDE_{12}
Ours+Multi	1.44/3.15	0.51/0.85
Ours+Single	1.44/3.16	0.53/0.92

(b) Argoverse Forecasting

Table 1: Trajectory generation from single mode and multiple modes

mated as

$$AADE(\mathcal{Y}_i) = \frac{1}{|\mathcal{Y}_i|} \sum_{\mathbf{Y} \in \mathcal{Y}_i} ADE(\hat{\mathbf{Y}}, \mathbf{Y}) \approx \frac{1}{K} \sum_{k=1}^{K} ADE(\hat{\mathbf{Y}}_k, \mathbf{Y}_r) \tag{3}$$

where $K = |\mathcal{Y}_i|$ is large enough. Here $\mathbf{Y}_r$ and $\hat{\mathbf{Y}}_k$ are the most representative trajectory in $\mathcal{Y}_i$ and its k-th prediction, respectively. The last term of Eqn. 3 is the average quality of the K trajectories generated for $\mathbf{Y}_r$. Consequently, the ADE_1 metric represents the average quality. The same derivation can be applied for the FDE_1 metric.

D Trajectory Generation from The Most Prominent Mode

We show in Table 1 the ADE and FDE performance of our forecasting model when K trajectories are generated from the most prominent mode only. In the table, **Ours+Multi** denotes the inference method that generates K future trajectories from the M modes. This method is the same as that described in the paper. **Ours+Single** denotes the inference method that generates K future trajectories from the most prominent mode, which is identified by the weight

distribution $\{w_m\}_{m=1}^{M}$. We can observe from the table that the best quality ($K \geq 12$) is degraded when the trajectories are generated from the most prominent mode only. On the other hand, **Ours**+**Single** shows nearly the same average quality performance as **Ours**+**Multi**. These are very natural results. When sampling a single future trajectory, the most prominent mode will be chosen for the sampling. Therefore, **Ours**+**Multi** and **Ours**+**Single** will show the same performance. On the other hand, when sampling multiple future trajectories, the trajectories generated by **Ours**+**Multi** will better reflect the true future trajectory distribution. Therefore, **Ours**+**Multi** will outperforms **Ours**+**Single** in terms of the best quality.

E Implementation Details

E.1 Candidate Lanes Acquisition

We identify $M = 10$ lane candidates for each target vehicle based on the method proposed in [5, 8, 10]. The lane segments within the search radius (10 meters) from the current position of the vehicle are first found. Next, lane candidates 80 meters long in the vehicle's heading direction are obtained by attaching the preceding and succeeding lane segments based on lane connectivity information provided by the HD maps. The set of coordinate points for the lane candidates is re-sampled such that any two adjacent coordinate points have equal distance (1 meter). The ground-truth lane on which the target vehicle has moved during the future timesteps is identified by the Euclidean distance between the ground-truth future trajectory and the lane candidates. If the number of the identified lane candidates is less than M, we add fake lane candidates with coordinate points of (0, 0). If the number is greater than M, $M-1$ randomly selected lanes and the ground-truth lane are used.

E.2 Details of Our Implementation

Preprocessing: Let $\mathbf{p}_i^t = (p_x^t, p_y^t)$ denote the position of the vehicle V_i at t. The speed s (meter per second) and heading h (radian) of the vehicle at t are calculated as follows:

$$s = \psi \sqrt{(p_x^t - p_x^{t-1})^2 + (p_y^t - p_y^{t-1})^2}, \tag{4}$$

$$h = \arctan\left(\frac{p_y^t - p_y^{t-1}}{p_x^t - p_x^{t-1}}\right), \tag{5}$$

where ψ is the sampling rate. Let $\mathbf{l}_f^m$ denote the coordinate of the f-th point of the lane $\mathbf{L}^m$. The tangent vector $\mathbf{v}_f = (v_{f,x}, v_{f,y})$ and its direction $d_{\mathbf{v}_f}$ at the point are calculated as follows:

$$\mathbf{v}_f = \mathbf{l}_f^m - \mathbf{l}_{f-1}^m \tag{6}$$

$$d_{\mathbf{v}_f} = \arctan\left(\frac{v_{f,y} - v_{f-1,y}}{v_{f,x} - v_{f-1,x}}\right). \tag{7}$$

Feature Extraction Module: The positional data $\mathbf{X}_i$, $\mathbf{Y}_i$, and $\mathbf{L}^m$ are first preprocessed by the method proposed in this paper. Next, the data are embedded by single-layer MLPs followed by ReLU activation. The MLPs for $\mathbf{X}_i$ and $\mathbf{Y}_i$ take as input a 4-dimensional vector and output a 16-dimensional vector. The MLP for $\mathbf{L}^m$ takes as input a 5-dimensional vector and outputs a 64-dimensional vector. Finally, the embedded sequential vectors are encoded by LSTM networks. The final hidden states of the LSTM networks are used for the final encodings. The hidden state size of the LSTM networks for $\mathbf{X}_i$ and $\mathbf{Y}_i$ is 16. The hidden state size for $\mathbf{L}^m$ is 64.

Scene Context Extraction Module: The attention operation between $\tilde{\mathbf{X}}_i$ and $\tilde{\mathbf{L}}^{(1:M)}$ for the context vector $\mathbf{a}_i^m$ is based on [1]. The context vector $\mathbf{b}_i^m$ is calculated as follows: The messages coming to the node V_i are first calculated by a single-layer MLP followed by ReLU activation, which takes as input a 34-dimensional vector and outputs a 16-dimensional vector, and then summarized by the sum operation. The summarized message is used to update the hidden state of the node. To update the hidden state, we use a GRU cell, which takes as input a 16-dimensional vector and outputs a 16-dimensional hidden state vector. After the one round of the message passing, $\mathbf{b}_i^m$ is obtained by summing the hidden states of the neighboring nodes.

Mode Selection Network: Ten lane-level scene context vectors $\{\mathbf{c}_i^m\}$ are first embedded by a single-layer MLP followed by ReLU activation, which takes as input a 160-dimensional vector and outputs a 64-dimension vector. The embedded vectors are then concatenated and used as input to a single-layer MLP, which takes as input a 640-dimensional vector and outputs a 10-dimension vector, to obtain the latent vector $\mathbf{z}_h$.

Encoder and Prior: The encoder produces the mean and variance vectors from the lane-level scene context vector $\mathbf{c}_i^m$ and the positional data encoding $\tilde{\mathbf{Y}}_i$. We use two two-layer MLPs for the mean and variance, respectively. The first layers of the MLPs take as input a 178-dimensional vector and output a 64-dimensional vector. The second layers take as input a 64-dimensional vector and output a 16-dimensional vector. The prior produces the mean and variance vectors from $\mathbf{c}_i^m$. The networks for the prior have the same structure as those for the encoder except that the first layers of the MLPs take as input a 160-dimensional vector. Finally note that we use ReLU activation for the first layers of the MLPs.

Decoder: To produce the next position $\hat{\mathbf{p}}_i^{t+1}$, the current position $\hat{\mathbf{p}}_i^t$ is first embedded by a single-layer MLP followed by ReLU activation, which takes as input a 2-dimensional vector and output a 16-dimensional vector. Next, $\mathbf{c}_i^m$, $\mathbf{z}_l$, and the embedding are concatenated and used as input to an LSTM network, which takes as input a 192-dimensional vector and outputs a 128-dimensional

hidden state vector, to update the hidden state vector. The next position is obtained by a single-layer MLP, which takes as input a 128-dimensional vector and outputs a 2-dimensional vector.

Discriminator: The positional data $[\mathbf{Y}_i; \Delta\mathbf{Y}_i]$ is first embedded by a single-layer MLP followed by ReLU activation, which takes as input a 4-dimensional vector and outputs a 16-dimensional vector. The embedded sequential data is then encoded by an LSTM network, which takes as input 16-dimensional sequential vectors and outputs 16-dimensional sequential hidden state vectors. The future encoding and lane encoding $\tilde{\mathbf{L}}_m$ are then used as input to a single-layer MLP to produce a scalar value. The MLP takes as input an 80-dimensional vector.

Training: Adam optimizer [9] is used for the optimization with initial learning rates of 10^{-4} (nuScenes) and 5×10^{-4} (Argoverse Forecasting) and batch size of 8 for 100 (nuScenes) and 50 (Argoverse Forecasting) epochs. We evaluate the prediction performance after every three consecutive training epochs by using the validation samples in the training dataset. Whenever the prediction performance improves over the past, we save the model's network parameters. During the training, we use a cyclical annealing schedule [7] for β.

E.3 Details of Ablation Study

We describe the details of the ablation study shown in section 4.3 of the paper. For **M1**, we do not use the positional data preprocessing (PDP), VLI, V2I, and GAN regularization proposed in the paper. As a result, the lane-level scene context vector $\mathbf{c}_i^m$ is defined as $\mathbf{c}_i^m = [\tilde{\mathbf{X}}_i; \tilde{\mathbf{L}}^m]$. For **M3**, we use the VLI so that $\mathbf{c}_i^m = [\tilde{\mathbf{X}}_i; \mathbf{a}_i^m]$. Finally, $\mathbf{c}_i^m = [\tilde{\mathbf{X}}_i; \mathbf{a}_i^m; \mathbf{b}_i^m]$ is used for **M4**, which employ the VLI and V2I.

E.4 Details of Baselines

We describe the details of the baseline models shown in Figure 3 of the paper. For the figure, we exclude the scene context extraction module and discriminator to show how helpful the introduction of the hierarchical latent structure would be for the mitigation of mode blur. Finally, note that the trajectories depicted in Figure 3-(a) of the paper is generated from **M2**.

Baseline: We train a generative model with a latent variable to model the trajectory distribution. One scene context vector $\mathbf{c}_i$ that condenses the information about all the modes of the distribution is first calculated as follows:

$$\mathbf{c}_i = [\tilde{\mathbf{X}}_i; \tilde{\mathbf{L}}^{ATT}], \tag{8}$$

where $\tilde{\mathbf{L}}^{ATT}$ is the result of the attention operation [1] between $\tilde{\mathbf{X}}_i$ and $\tilde{\mathbf{L}}^{(1:M)}$. $\mathbf{c}_i$ is then used as input to the encoder, prior, and decoder.

Baseline+BOM: We train **Baseline** with the best-of-many (BOM) sample objective [2]. During the training, we let the model generate five trajectories per vehicle and select the trajectory with the minimum ADE out of the five for the $L2$-distance loss calculation.

Baseline+NF: We train **Baseline** with normalizing flows (NF) [12]. We apply ten planar flow operations to a random vector that follows the normal distribution to obtain the final latent variable.

References

1. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: Int. Conf. on Learn. Represent. (2015)
2. Bhattacharyya, A., Schiele, B., Fritz, M.: Accurate and diverse sampling of sequences based on a best-of-many sample objective. In: IEEE Conf. Comput. Vis. Pattern Recog. (2018)
3. Casas, S., Gulino, C., Suo, S., Luo, K., Liao, R., Urtasun, R.: Implicit latent variable model for scene-consistent motion forecasting. In: Eur. Conf. Comput. Vis. (2020)
4. Casas, S., Gulino, C., Suo, S., Urtasun, R.: The importance of prior knowledge in precise multimodal prediction. In: Int. Conf. Intell. Robots Syst. (2020)
5. Chang, M.F., Lambert, J., Sangkloy, P., Singh, J., Bak, S., Hartnett, A., Wang, D., Carr, P., Lucey, S., Ramanan, D., Hays, J.: Argoverse: 3d tracking and forecasting with rich maps. In: IEEE Conf. Comput. Vis. Pattern Recog. (2019)
6. Cui, A., Sadat, A., Casas, S., Liao, R., Urtasun, R.: Lookout: diverse multi-future prediction and planning for self-driving. In: Int. Conf. Comput. Vis. (2021)
7. Fu, H., Li, C., Liu, X., Gao, J., Celikyilmaz, A., Carin, L.: Cyclical annealing schedule: A simple approach to mitigating kl vanishing. In: NAACL (2019)
8. Kim, B., Park, S.H., Lee, S., Khoshimjonov, E., Kum, D., Kim, J., Kim, J.S., Choi, J.W.: Lapred: lane-aware prediction of multi-modal future trajectories of dynamic agents. In: IEEE Conf. Comput. Vis. Pattern Recog. (2021)
9. Kingma, D.P., Ba, L.J.: Adam: a method for stochastic optimization. In: Int. Conf. on Learn. Represent. (2015)
10. Narayanan, S., Moslemi, R., Pittaluga, F., Liu, B., Chandraker, M.: Divide-and-conquer for lane-aware diverse trajectory prediction. In: IEEE Conf. Comput. Vis. Pattern Recog. (2021)
11. P-Minh, T., Grigore, E.C., Boulton, F.A., Beijbom, O., Wolff, E.M.: Covernet: multimodal behavior prediction using trajectory sets. In: IEEE Conf. Comput. Vis. Pattern Recog. (2020)
12. Rezende, D.J., Mohamad, S.: Variational inference with normalizing flows. In: Int. Conf. on Mach. Learn. (2015)