HOPE: Hierarchical Spatial-temporal Network for Occupancy Flow Prediction

Yihan Hu, Wenxin Shao, Bo Jiang, Jiajie Chen, Siqi Chai
Zhening Yang, Jingyu Qian, Helong Zhou, Qiang Liu
Horizon Robotics
{yihan.hu96, nemonswx}@gmail.com

Abstract

In this report, we introduce our solution to the Occupancy and Flow Prediction challenge in the Waymo Open Dataset Challenges at CVPR 2022. We have developed a novel hierarchical spatial-temporal network featured with spatial-temporal encoders, a multi-scale aggregator enriched with latent variables, and a recursive hierarchical 3D decoder. We use multiple losses including focal loss and modified flow trace loss to efficiently guide the training process. Our method achieves a Flow-Grounded Occupancy AUC of 0.8389 and outperforms all the other teams on the leaderboard.

1. Introduction

Since 2020, the Waymo Open Dataset [6, 24] has been providing the research communities with high-quality data collected from both LiDAR and camera sensors in real self-driving scenarios, which have enabled a lot of new exciting research. At CVPR 2022, the Waymo Open Dataset Challenge introduces a brand new challenge named "Occupancy and Flow Prediction" by expanding original Motion Datasets [6]. This new form of representation [11, 19] mitigates the shortcomings of the existing representation such as trajectory sets and occupancy. By predicting flow fields and occupancy jointly, the motion and location probability of all agents in the scene can be captured simultaneously. More specifically, given one-second history of agents in a scene, the algorithm is required to forecast all vehicles' occupancy and flow fields in 8 seconds. The evaluation metric is calculated using Flow-Grounded Occupancy, which measures the quality of predicted occupancy and flow jointly. The algorithm that achieves best AUC score with Flow-Grounded Occupancy wins the challenge.

In this report, we propose a hierarchical spatial-temporal network named *HOPE*, composed of spatial-temporal encoders, a multi-scale aggregator enriched with latent variables, and a hierarchical 3D recursive decoder. Our model takes the past 10 frames' and the current frame's as input. We enrich the history with a set of rasterized agents' dynamic states with the scene's static information. Multiple modified losses are used to guide the training convergence. For better result, we have also utilized Stochastic Weights Averaging (SWA), test-time augmentation (TTA) and model ensembling techniques to further refine the predictions.

2. Methods

We divide our model into three parts. The encoder generates multi-scale features from the spatial-temporal inputs. The aggregator interacts and fuses the features at different scales of the encoders. Then, the condensed features are decoded into temporal-coherently predicted waypoints by a spatial-temporal decoder. Fig. 1 shows an overview structure of our model. In this section, we first introduce our input representation, and then elaborate the structure of our model.

2.1. Inputs Representation

For a given scene, our input $I_t = (D_t, S)$ contains both the dynamic features and the static features. The dynamic features $D_t = (O_t, a_t)$ include the occupancy information O_t as well as the attributes of an agent a_t such as bounding box's width, height, length, z , velocity x, velocity y, and speed. Static features $S = s_{1,...,N}$ include road map information and each channel represents a different static element (e.g overall road map, middle line, road edge, traffic light). There are 29 different road features thus $N = 28$. We encode past 10 scenes and the current scene for a total of 11 scenes and we use a rasterized bird-eye-view(BEV) as the representation. To be specific, the shape of O_t, a_t and S are $H \times W \times 3$(for all 3 classes), $H \times W \times 7$ and $H \times W \times 29$.

For 2D encoders, the dynamic features are concatenated along the temporal dimension, followed by the static features, denoted as $I = (D_{1,...,11}, S)$. We merge the occupancy of bicycle and pedestrian classes, thus the channel of occupancy is 22. For agent attributes, since bounding box's width, height and length features are also static, we merge them across time, thus the channel of agent attributes

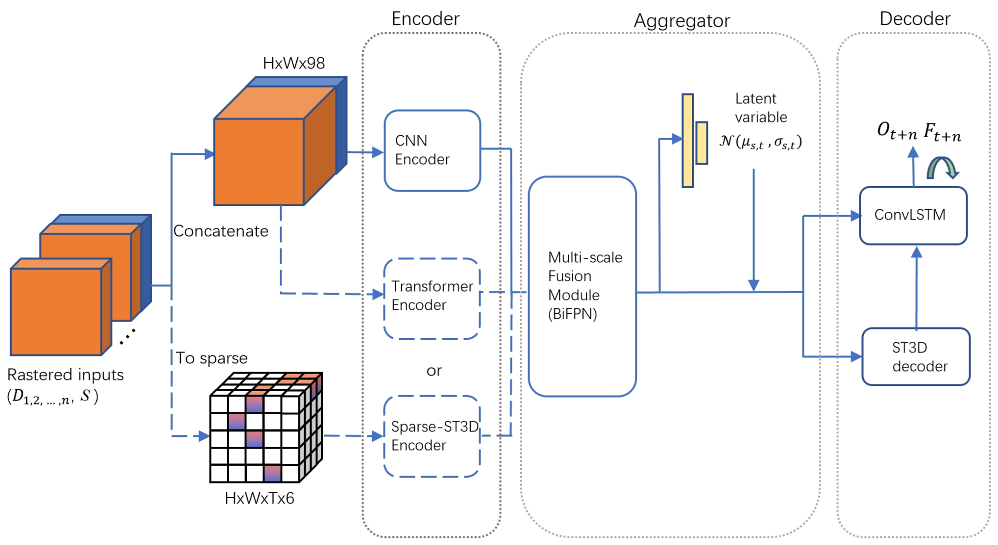


Figure 1. Overall structure of our model. The inputs include the dynamic features (e.g. agents' occupancy) marked in orange, and the static features (e.g. map states) marked in blue. For the inputs to the 3d sparse encoder, each cube represents both the dynamic features and the static features marked in orange and blue respectively. To reduce the computational cost, we only choose either the transformer encoder or the sparse ST3D encoder to fuse with our CNN encoder.

is $4 \times 11 + 3 = 47$. With 29 channels of road features S, the total input channel is 98.

For 3D encoder, we encode the time as an additional dimension, as $I = I_{1,...,11}$ for both the dynamic features and the static features, which is achieved by duplicating the static features across the time dimension. We also simplify both the dynamic features and the static features: for the dynamic features, only the occupancy information of the 3 classes is encoded; and for the static features, we merge all road map information into 3 channels: overall road center, overall road edge, and all other road information (e.g. stop sign, crosswalk, and speed bump). Thus, each feature is 6 dimensional and the final input shape before sparsify is $H \times W \times T \times 6$.

2.2. Model Structure

2.2.1 Encoder

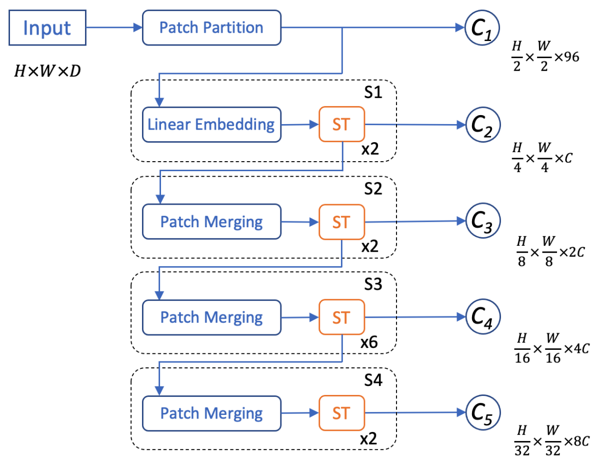
We design and combine 3 different encoders for the task: a 2D CNN encoder, a Transformer-based encoder, and a spatial-temporal sparse 3D CNN encoder. Then the encoded features by different encoders are concatenated and fused at each scale.

CNN Encoder We use RegNet [22] as our 2D CNN encoder. RegNet is a well-designed backbone featured with high efficiency. In our implementation, we have utilized RegNetX-32GF. The encoder generates multi-scale features C_1, C_2, C_3, C_4, C_5, where C_i denotes the feature of spatial size $\frac{H}{2^i} \times \frac{W}{2^i}$ as the input of the aggregator.

SwinTransformer Recently, Transformer-based models have achieved great success on vision tasks. Specifically, the SwinTransformer [15, 16] has become one of the state-of-the-art models on many dense prediction tasks such as detection and segmentation. The SwinTransformer is an excellent choice for our task for three reasons: First, the occupancy flow prediction is also a dense prediction task; Second, the attention mechanism is widely used in motion prediction tasks [4,7,20,27] as it matches well with the real-world driving scenarios, where interactions occur between agents as well as between agents and roads. Especially, the window attention mechanism greatly reduces the computational cost when the attention ranges increase. Third, the huge amount of data by Waymo Open Dataset provides an excellent and necessary condition for the SwinTransformer to converge.

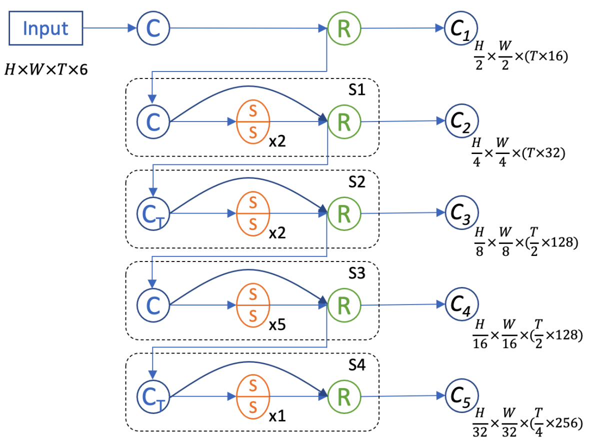
In practice, we use SwinV2-T as one of the encoder networks. Similar as the CNN encoder, it also generates multi-scale features C_1, C_2, C_3, C_4, C_5, thus these features can be concatenated with the features from other encoders of the same spatial sizes. Note that we use the ImageNet1K [3] pretrained model with a window size of 16, and we only change the patch size from 4 to 2 and leave other hyper parameters unchanged compared with the original SwinV2-T. Figure 2 is an overview of the SwinTransformer module in our task.

Spatial-temporal 3D Encoder To better exploit the temporal or inter-frame information, we have designed a



Input
$H\times W\times D$
Patch Partition
C_1
$\frac{H}{2}\times\frac{W}{2}\times 96$
S1
Linear Embedding
ST
x2
C_2
$\frac{H}{4}\times\frac{W}{4}\times C$
S2
Patch Merging
ST
x2
C_3
$\frac{H}{8}\times\frac{W}{8}\times 2C$
S3
Patch Merging
ST
x6
C_4
$\frac{H}{16}\times\frac{W}{16}\times 4C$
S4
Patch Merging
ST
x2
C_5
$\frac{H}{32}\times\frac{W}{32}\times 8C$

Figure 2. The structure of the SwinTransformer encoder. 'ST' stands for Swin Transformer Block. H, W, D, C are the input height, width, channel and embedding channel, respectively. C_1-C_5 are the generated multi-scale features.



Input
$H\times W\times T\times 6$
C
R
C_1
$\frac{H}{2}\times\frac{W}{2}\times(T\times 16)$
S1
S
x2
C_2
$\frac{H}{4}\times\frac{W}{4}\times(T\times 32)$
S2
C_T
x2
C_3
$\frac{H}{8}\times\frac{W}{8}\times(\frac{T}{2}\times 128)$
S3
x5
C_4
$\frac{H}{16}\times\frac{W}{16}\times(\frac{T}{2}\times 128)$
S4
x1
C_5
$\frac{H}{32}\times\frac{W}{32}\times(\frac{T}{4}\times 256)$

Figure 3. The structure of the 5-stage 3D encoder. 'C' represents 3D sparse convolutional layer that downsamples H and W dimension, 'C_T' represents 3D sparse convolutional layer that downsamples H, W, and T dimension, 'S' represents submanifold sparse convolutional layer (two stacked 'S' means two such layers in a row), and 'R' represents reshape operation that merges time dimension and feature dimension.

spatial-temporal 3D encoder to capture temporal consistency. Since the rasterized input data has high sparsity, we have used 3D sparse convolutional layer and submanifold sparse convolutional layer [8] to both accelerate the forward speed and better capture the input features across the spatial and temporal dimensions. We construct a 5-stage backbone, where each stage downsamples the spatial dimensions by 2. For the temporal dimension, it is only downsampled at stage 2 and 4. The output of each stage is gathered and the temporal and feature dimensions of each output are merged as the final feature dimension. Figure 3 shows the detailed structure of the sparse encoder.

2.2.2 Aggregator

As shown in Fig. 4, the aggregator is composed of a set of stacked BiFPN layers and a multi-scale latent variable module, which aggregates the spatial features at different scales and fuses them with the latent information.

BiFPN Following EfficientDet [19, 25], the multi-scale, temporal-condensed features are fused and interacted in a bidirectional manner using BiFPN layers.

Latent Variable Inspired by FIERY [11], we model the inherent stochastic future using latent variables by a conditional variational approach. However, the latent code in FIERY only models the entire scene, mainly the ego vehicle, which is not enough for our task. To expand it to all of the agents in the scene, we model the latent variables as Gaussian distributions in multi-scale bird-eye-view(BEV). During training, we sample the latent code from the current distribution at every location and scale, and take the mean of the distribution during inference. The latent feature maps ($\Phi_{s,t}$ with the shape $H_s \times W_s \times C$) and the vector ($\phi_{global,t}$ with the shape $1 \times 1 \times C$) are then expanded, concatenated and fused with the multi-scale features. Also, to enforce the consistency of the predicted and observed future distribution, the Kullback-Leibler divergence loss is adopted during training as described in 2.3

2.2.3 Decoder

Different from image segmentation task which only considers spatial information, the occupancy and flow prediction take both of the spatial and temporal information into account. Thus, naively applying the decoders from segmentation task is not the optimal choice. Recently, spatial-temporal prediction has gained more advancements, such as in video prediction [21]. Inspired by the previous work [9–11, 23, 26], we design a novel hierarchical spatial-temporal decoder as shown in Fig. 6. Different from the previous video prediction task, the occupancy and flow prediction has much larger spatial dimension (256x256) and a longer forecasting time horizon (8 second), which implies long range dependence in both spatial and temporal domain. Thus, widely used ConvGRU [1] and ConvLSTM [23] are not an optimal choice since they have small receptive field and relatively large computational cost due to their recursive nature. To enlarge our receptive field without sacrificing the efficiency, we design stacked dilated 3D convolutional bottleneck structures as shown in Fig. 5. The dilation [2] makes our receptive field extremely large with almost the same computational complexity. Also, the grouped bottleneck structures with skip connections further boost our model's speed and efficiency. To incorporate multi-scale features, the stacked bottlenecks are then combined in a FPN manner as shown in Fig. 6. Since our input multi-scale features from the aggregator are condensed in temporal domain (i.e P_2, P_3, ..., P_6), we further adopt transposed 3D convolutions in the bottleneck structures which gradually unroll the temporal domain.

To incorporate fine-grained features efficiently, we adopt a single ConvLSTM layer at the top. As shown in Fig. 6, the ConvLSTM layer takes largest scale output by the aggregator (i.e P_1) as the hidden and the cell states. As a refinement, the ConvLSTM layer recursively takes the temporal-unrolled coarse output by the 3D-FPN structures as input. Then, the unrolled features are sent to the shared heads which are finally decoded as the observed/occluded occupancy (O_{t+n}) and the flow prediction (F_{t+n}).

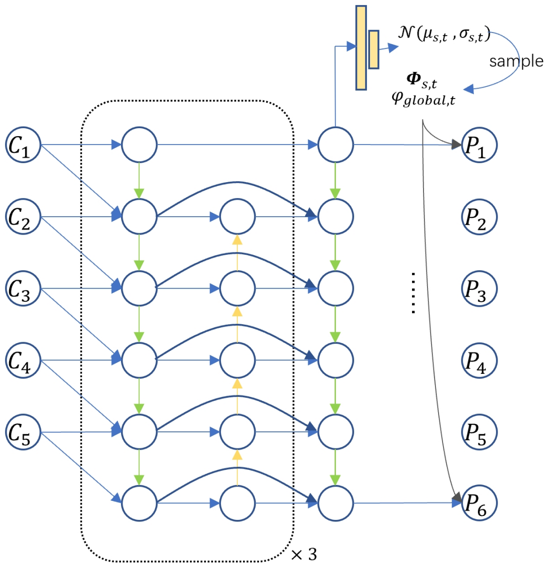


Figure 4. The structure of the aggregator. C_1, ..., C_5 are the fused features from 5 different scales. $\Phi_{s,t}$ and $\phi_{global,t}$ are the latent BEV map and the global latent vector respectively, which are sampled from the predicted distributions at scale s. The latent map are then concatenated with the BiFPN features as the aggregated output P_1, ..., P_6.

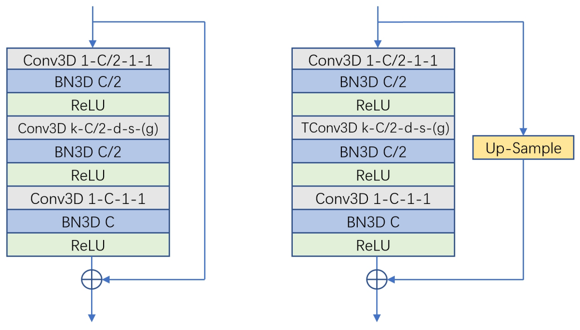


Figure 5. The structure of the 3D convolutional bottleneck block(Conv3D-BN). "Conv3D" stands for 3D convolutional layer and "TConv3D" stands for transpose 3D convolutional layer. The format of the layer setting follows "kernel size-channels-dilations-strides-(groups)", *i.e.* k-C-d-s-(g). The kernel size, dilations, and strides are tuples following (T, H, W), where T represents the time domain, and H, W represent the spatial domain. To match the output shape of transpose conv3d layer, a temporal up-sample layer is used at the skip connection.

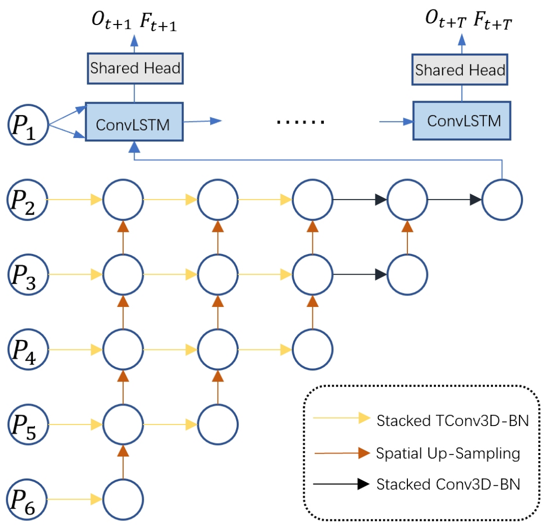


Figure 6. The structure of the Spatial Temporal Decoder. "Conv3D-BN" and "TConv3D-BN" stand for the regular and transposed 3D convolutional bottleneck block respectively.

2.3. Losses

Occupancy Loss Focal loss [14] is adopted to supervise both the observed and occluded occupancy prediction, as shown in Eq. 1.

$$\mathcal{L}_O^{b,c} = \sum_{t=1}^{T_{\text{pred}}} \sum_{x=0}^{w-1} \sum_{y=0}^{h-1} \mathcal{F}\left(O_t^{b,c}(x,y), \tilde{O}_t^{b,c}(x,y)\right) \quad (1)$$

$$\mathcal{L}_{occ} = \lambda_O^b \mathcal{L}_O^b + \lambda_O^c \mathcal{L}_O^c \quad (2)$$

where O^b is observed occupancy, O^c is occluded occupancy, O_t is the predicted occupancy and $\tilde{O}_t$ is the ground truth occupancy at time step t.

Flow Loss Similar to Mahjourian et al. [19], we apply smooth L1 loss for the flow regression. To decouple the flow prediction and occupancy prediction tasks, the loss is further weighted by the ground truth occupancy $\tilde{O}_t$.

Traced Loss We implement the traced loss as described in Mahjourian et al. [19] with some modifications. Instead of using cross-entropy loss alone, we mix the focal loss and the cross-entropy loss to further boost the Flow-Grounded Occupancy AUC metric. Also, to be consistent with the Flow-Grounded Occupancy AUC metric, instead of recursively applying the warping process on current occupancy O_0, we directly warp the ground truth O_{t-1} at the previous time step with the predicted flow F_t , as shown in Eq. 3.

$$\mathcal{W}_t = F_t \circ \tilde{O}_{t-1} \tag{3}$$

Then cross entropy loss and focal loss are applied jointly as the final traced loss, as shown in Eq. 4

$$\mathcal{L}_{\text{traced}} = \sum_{t=1}^{T_{\text{pred}}} \sum_{x=0}^{w-1} \sum_{y=0}^{h-1} \lambda_{ce} \mathcal{H}\left(\mathcal{W}_t(x,y) O_t(x,y), \tilde{O}_t(x,y)\right) + \lambda_f \mathcal{F}\left(\mathcal{W}_t(x,y) O_t(x,y), \tilde{O}_t(x,y)\right) \tag{4}$$

where $\mathcal{H}$ is the cross entropy loss.

Probabilistic Loss Following FIERY [11], we use Kullback-Leibler divergence to enforce the consistency between the predicted future distributions and the observed future distributions. Different from FIERY, we average the KL-divergence loss over every pixel and scale, as shown in Eq. 5 and Eq. 6.

$$L_{\text{prob}} = \frac{1}{Nhw} \sum_{n=0}^{N} \sum_{x=0}^{w-1} \sum_{y=0}^{h-1} \lambda_n D_n \tag{5}$$

$$D_n = D_{\text{KL}}\left(F\left(\cdot \mid s_{n,t}, y_{t+1}, \ldots, y_{t+T}\right) \| P\left(\cdot \mid s_{n,t}\right)\right) \tag{6}$$

where N is number of scales, $s_{n,t}$ is the n-th output feature map, $y_{t+1}, ..., y_{t+T}$ are ground truth occupancy and flows. In our experiment, we set equal weight to each scale, i.e $\lambda_s = 1$.

We use the weighted sum of all losses as the final loss:

$$\mathcal{L} = \lambda_{\text{occ}} \mathcal{L}_{\text{occ}} + \lambda_{\text{flow}} \mathcal{L}_{\text{flow}} + \lambda_{\text{traced}} \mathcal{L}_{\text{traced}} + \lambda_{\text{prob}} \mathcal{L}_{\text{prob}} \tag{7}$$

3. Experiments

3.1. Experiment Settings

We generate the rasterize input within a $120m \times 120m$ region at a resolution of 0.156m/pixel, thus the input spatial size is 768×768. To match the output resolution with the evaluation range, the mutliscale feature maps are cropped with a 2/3 ratio to keep the $80m \times 80m$ region before the decoder. We use AdamW optimizer [18] and cosine annealing policy [17] with an initial learning rate of 2.5×10^{-4}. The weight decay is set to 0.01. We set the loss coefficient $\lambda_{\text{occ}} = 500$, $\lambda_{\text{flow}} = 1$, $\lambda_{\text{traced}} = 500$ and $\lambda_{\text{prob}} = 1$. For the occupancy loss, weights between the observed and occluded occupancy are equal, i.e $\lambda_O^b = \lambda_O^c = 1$. For the traced loss, weights between the cross entropy and the focal loss are equal, i.e $\lambda_{ce} = \lambda_f = 1$.

Model Variations and Ensembling We train two model variants: HOPE-Swin and HOPE-3D. HOPE-Swin fuses RegNet and SwinV2 encoder while HOPE-3D fuses RegNet and Sparse encoder. Both models are trained on the training and validation datasets. HOPE-Swin and HOPE-3D are trained for 4 and 2 epochs respectively.

For both models, we use Stochastic Weights Averaging (SWA) [12] to further enhance the training. We train each model for one additional epoch using decreased learning rate and greedily average the weights.

3.2. Test-Time Augmentation and Model Ensembling

We also boost predictions through Test-Time Augmentation (TTA) [5, 13], by rotating the raster inputs by π, and ensembling by taking the weighted average with the original predictions. Note that we only apply TTA on flow predictions, keeping occupancy predictions as the original. The weight of the TTA prediction is 0.25. The effectiveness of TTA is shown in Tab. 2.

After the TTA, we ensemble the output of the two model variants by taking a weighted averaging as the final result.

4. Results

4.1. Performance on the Test Set

The final results on the official Occupancy and Flow Prediction challenge leaderboard are shown in Tab. 1. Based on the primary metric, Flow-Grounded Occupancy AUC, we outperform all the other teams. Fig. 7 shows the visualization of some of our predictions.

4.2. Ablation Studies

To verify the effectiveness of our approach, we have conducted a number of ablation studies as shown in Fig. 2.

Losses We replace the losses in our baseline model with the focal loss and the traced loss. The Flow-Grounded Occupancy AUC and Flow-Grounded Occupancy soft-IoU are boosted dramatically by replacing the cross-entropy loss with the focal loss [14], at some decrease of the Observed soft-IoU metric. Moreover, adding traced loss [19] further boosts Flow-Grounded Occupancy AUC and Flow-Grounded Occupancy soft-IoU by a large margin.

Additional Input As described in 2.1, besides the simple rasterized occupancy O_t, we further enrich the dynamic features $D_t = (O_t, a_t)$ by rasterizing the agents' attribute a_t(i.e bounding box's width, height, length, z, velocity x, velocity y and speed). Also, we expand the static feature S_t from a single one-hot layer to 28 layers, which includes lanes' and traffic states' information. By adding additional

Models	Observed Occupancy		Occluded Occupancy		Flow	Flow-Grounded Occupancy	
	AUC	Soft IoU	AUC	Soft IoU	EPE	AUC	Soft IoU
HorizonOccFlowPrediction(Ours)	**0.8033**	0.2349	0.1650	0.0169	3.6717	**0.8389**	**0.6328**
Look Around	0.8014	0.2336	0.1386	0.0285	**2.6191**	0.8246	0.5488
Temporal Query - Stable	0.7565	0.3934	0.1707	0.0404	3.3075	0.7784	0.4654
STrajNet	0.7514	0.4818	0.1610	0.0183	3.5867	0.7772	0.5551
VectorFlow	0.7548	**0.4884**	**0.1736**	**0.0448**	3.5827	0.7669	0.5298

Table 1. Top five submissions of the Occupancy and Flow Prediction challenge. For each team, only the best submission is kept. The results are evaluated on the test set. The Flow-Grounded Occupancy AUC (marked in grey) is used as primary metric.

focal loss	traced loss	enriched inputs	latent var.	ST decoder	+swin encoder	+ST3D encoder	TTA	Flow-Grounded		Observed Occupancy		Flow
								AUC	Soft-IoU	AUC	Soft-IoU	EPE
								0.7492	0.4905	0.7193	0.4403	4.4692
✓								0.7693	0.5109	0.7133	0.1924	4.2966
✓	✓							0.7739	0.5344	0.6792	0.1503	4.7153
✓	✓	✓						0.7862	0.5511	0.7053	0.1610	4.3068
✓	✓	✓	✓					0.7850	0.5523	0.7065	0.1655	4.3146
✓	✓	✓	✓	✓				0.7956	0.5580	0.7231	0.1466	4.1320
✓	✓	✓	✓	✓	✓			0.7972	0.5627	0.7258	0.1619	4.0951
✓	✓	✓	✓	✓		✓		0.7983	0.5647	0.7262	0.1611	4.0433
✓	✓	✓	✓	✓		✓	✓	0.8023	0.5788	0.7262	0.1611	3.9840

Table 2. The detailed ablation study. All the experiments are conducted on a 1/10 training set for 5 epochs and evaluated on the all validation set with the scenario IDs provided by WOD. The rasterized inputs are fixed at a range of 80mx80m. The metrics are evaluated under our customized environment, which is slightly different from the online evaluation server. For WOD2022 challenge, the "Flow-Grounded AUC" (marked in grey) is used as primary metric.

information, we observed significant improvements among all metrics.

Probabilistic Modeling Adding multi-scale latent variable slightly improves the Flow-Grounded Occupancy soft-IoU, Observed AUC, and Observed soft-IoU with the cost of a slight drop in Flow-Grounded Occupancy AUC.

Spatial Temporal Decoder Instead of predicting all 8 waypoints at the same time, we add a spatial temporal decoder and predict 8 future states recursively as described in 2.2.3. We have observed significant improvements in all metrics, showing the effectiveness of decoupling the spatial and temporal information when decoding.

Extra Encoder To further enhance our encoded features, we have designed two variant models: One by adding Swin-Transformer encoder 2.2.1 and the other by adding ST3D encoder 2.2.1. We can observe in Tab. 2 that fusing both encoders outperforms the single CNN encoder. Moreover, we can further benefit from the two variants of the model by using model ensemble technique.

5. Conclusion

In this report, we have shown our method for the occupancy flow prediction with a hierarchical spatial-temporal model named *HOPE*. This model is composed of multiple spatial-temporal encoders, a multi-scale latent code enriched aggregator, and a hierarchical 3D recursive decoder. In addition, we have redesigned multiple losses, including the focal loss and the flow trace loss. For further enhancements, we have adopted various ensemble techniques, i.e stochastic weights averaging and test time augmentation. Thanks to all these innovations, *HOPE* achieved the top accuracy in term of the Flow-Grounded Occupancy AUC for the "Occupancy and Flow Prediction" in the Waymo Open Dataset Challenges 2022.

References

[1] Nicolas Ballas, Li Yao, Chris Pal, and Aaron Courville. Delving deeper into convolutional networks for learning video representations. *arXiv preprint arXiv:1511.06432*, 2015. 3

[2] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017. 3

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 2

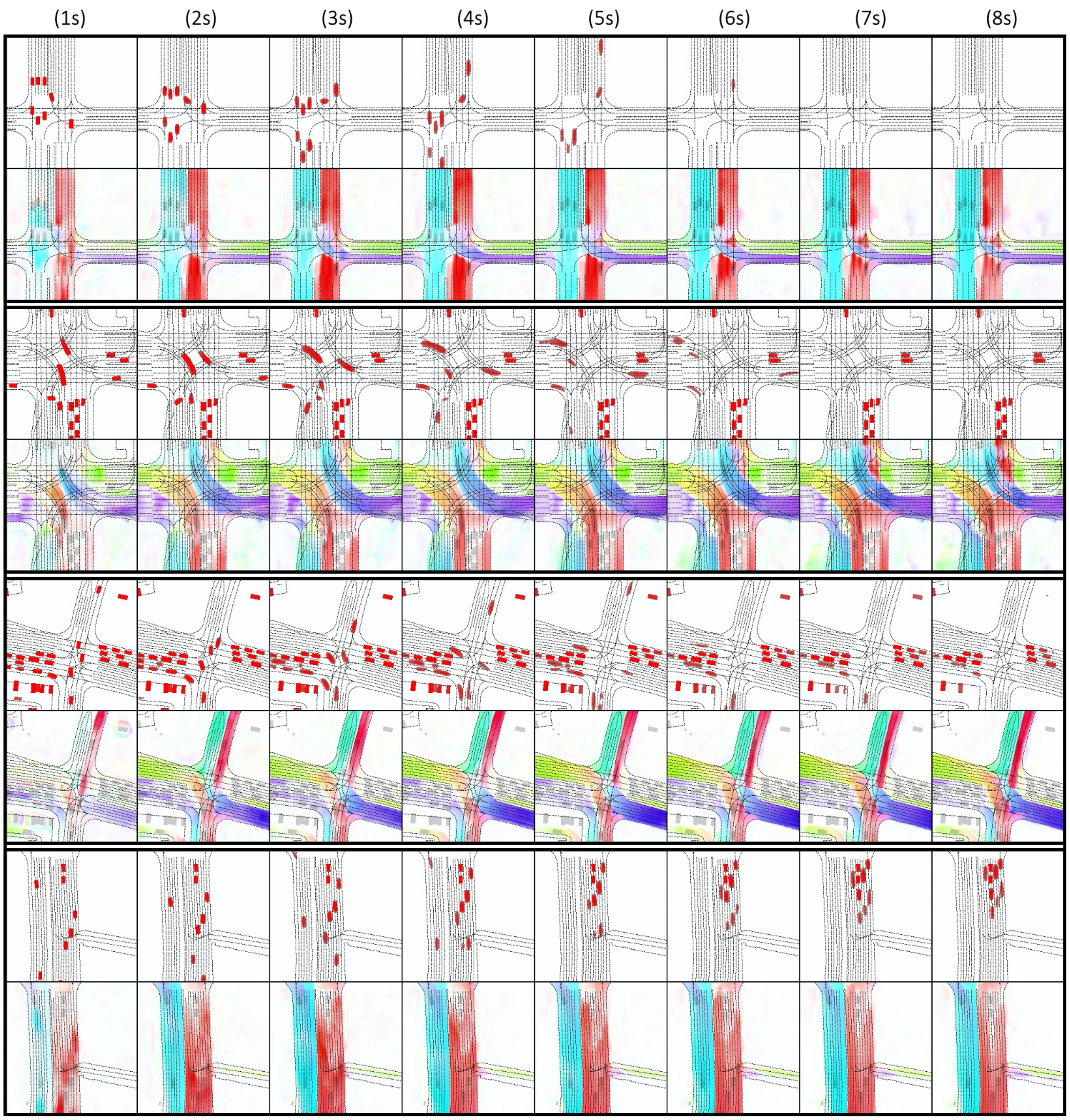


Figure 7. Prediction visualization of 4 scenes. For each scene, the first row shows the predicted occupancy with score threshold of 0.4 and the second row shows the predicted flow.

[4] Nachiket Deo, Eric Wolff, and Oscar Beijbom. Multimodal trajectory prediction conditioned on lane-graph traversals. In *5th Annual Conference on Robot Learning*, 2021. 2

[5] Zhuangzhuang Ding, Yihan Hu, Runzhou Ge, Li Huang, Sijia Chen, Yu Wang, and Jie Liao. 1st place solution for waymo open dataset challenge–3d detection and domain adaptation. *arXiv preprint arXiv:2006.15505*, 2020. 5

[6] Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles R. Qi, Yin Zhou, Zoey Yang, Aur'elien Chouard,

Pei Sun, Jiquan Ngiam, Vijay Vasudevan, Alexander McCauley, Jonathon Shlens, and Dragomir Anguelov. Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset. 1

[7] Jiyang Gao, Chen Sun, Hang Zhao, Yi Shen, Dragomir Anguelov, Congcong Li, and Cordelia Schmid. Vectornet: Encoding hd maps and agent dynamics from vectorized representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11525–11533, 2020. 2

[8] Benjamin Graham and Laurens van der Maaten. Submanifold sparse convolutional networks. *arXiv preprint arXiv:1706.01307*, 2017. 3

[9] Anthony Hu, Fergal Cotter, Nikhil Mohan, Corina Gurau, and Alex Kendall. Probabilistic future prediction for video scene understanding. In *European Conference on Computer Vision*, pages 767–785. Springer, 2020. 3

[10] Anthony Hu, Alex Kendall, and Roberto Cipolla. Learning a spatio-temporal embedding for video instance segmentation. *arXiv preprint arXiv:1912.08969*, 2019. 3

[11] Anthony Hu, Zak Murez, Nikhil Mohan, Sofía Dudas, Jeffrey Hawke, Vijay Badrinarayanan, Roberto Cipolla, and Alex Kendall. Fiery: Future instance prediction in bird's-eye view from surround monocular cameras. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15273–15282, 2021. 1, 3, 5

[12] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018. 5

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, may 2017. 5

[14] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 4, 5

[15] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, et al. Swin transformer v2: Scaling up capacity and resolution. *arXiv preprint arXiv:2111.09883*, 2021. 2

[16] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021. 2

[17] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 5

[18] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 5

[19] Reza Mahjourian, Jinkyu Kim, Yuning Chai, Mingxing Tan, Ben Sapp, and Dragomir Anguelov. Occupancy flow fields for motion forecasting in autonomous driving. *IEEE Robotics and Automation Letters*, 7(2):5639–5646, 2022. 1, 3, 4, 5

[20] Jiquan Ngiam, Benjamin Caine, Vijay Vasudevan, Zhengdong Zhang, Hao-Tien Lewis Chiang, Jeffrey Ling, Rebecca Roelofs, Alex Bewley, Chenxi Liu, Ashish Venugopal, et al. Scene transformer: A unified multi-task model for behavior prediction and planning. *arXiv e-prints*, pages arXiv–2106, 2021. 2

[21] Sergiu Oprea, Pablo Martinez-Gonzalez, Alberto Garcia-Garcia, John Alejandro Castro-Vargas, Sergio Orts-Escolano, Jose Garcia-Rodriguez, and Antonis Argyros. A review on deep learning techniques for video prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. 3

[22] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10428–10436, 2020. 2

[23] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. *Advances in neural information processing systems*, 28, 2015. 3

[24] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2446–2454, 2020. 1

[25] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10781–10790, 2020. 3

[26] Yunbo Wang, Lu Jiang, Ming-Hsuan Yang, Li-Jia Li, Mingsheng Long, and Li Fei-Fei. Eidetic 3d lstm: A model for video prediction and beyond. In *International conference on learning representations*, 2018. 3

[27] Hang Zhao, Jiyang Gao, Tian Lan, Chen Sun, Benjamin Sapp, Balakrishnan Varadarajan, Yue Shen, Yi Shen, Yuning Chai, Cordelia Schmid, et al. Tnt: Target-driven trajectory prediction. *arXiv preprint arXiv:2008.08294*, 2020. 2