VectorFlow: Combining Images and Vectors for Traffic Occupancy and Flow Prediction

Xin Huang[1], **Xiaoyu Tian**[2], **Junru Gu**[2], **Qiao Sun**[2], **Hang Zhao**[2]
[1]CSAIL, MIT [2]IIIS, Tsinghua University
xhuang@csail.mit.edu, hangzhao@mail.tsinghua.edu.cn

1 Approach Overview

In this challenge, our team explores the benefit of combining vectorized and rasterized representations in the task of occupancy and flow prediction, as proposed by [1]. Both representations have demonstrated great success in trajectory prediction benchmarks, yet few models combine them together. Our proposed model, *VectorFlow*, is a simple but effective approach that fuses both vectorized and rasterized representations of traffic context to predict observed occupancy, occluded occupancy, and flow.

2 Problem Formulation

In this work, we follow the problem proposed by the Waymo Occupancy and Flow Challenge[1]: Given one-second history of traffic agents in a scene and the scene context such as map [2], the objective is to predict i) future observed occupancy, ii) future occluded occupancy, and iii) future flow of all the vehicles in a scene over the next 8 waypoints, where each waypoint covers a one-second interval.

We process the input into a rasterized image and a set of vectors. We follow the challenge tutorial[2] to obtain the rasterized image, with respect to the local coordinate of the self-driving car (SDC). To obtain the vectorized input that is consistent with the rasterized image, we follow the same transformation by rotating and shifting the input agent and map coordinates with respect to SDC's local view.

3 Model Details

3.1 Encoder

The encoder includes two parts: a VGG-16 model [3] that encodes the rasterized representation, and a VectorNet [4] model that encodes the vectorized representation. We fuse the vectorized features with the features of the last two stages of VGG-16 by cross-attention modules. The fused features are upsampled to the original resolution as the input rasterized features by an FPN-style network. More details can be found in Fig. 1 and Sec. 4.3.

3.2 Decoder

The decoder is a single 2D convolution layer that maps the output of the encoder to the desired output, which includes a sequence of 8 grid maps representing the predicted occupancy and flow.

[1] https://waymo.com/open/challenges/2022/occupancy-flow-prediction-challenge/
[2] https://github.com/waymo-research/waymo-open-dataset/blob/master/tutorial/tutorial_occupancy_flow.ipynb

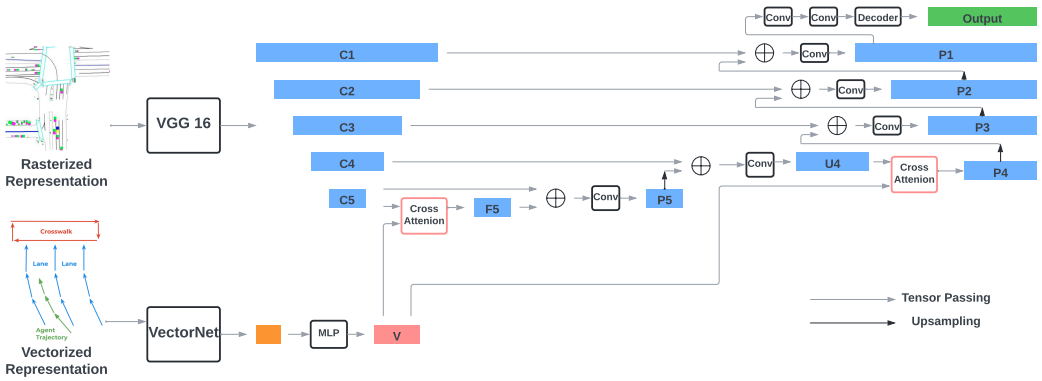

Figure 1: Overview of VectorFlow. More details can be found in Sec. 4.3. Representation images from [4].

3.3 Loss

We follow the loss function as in [1], including a cross entropy loss on observed occupancy prediction $\mathcal{L}_{O,b}$, a cross entropy loss on occluded occupancy prediction $\mathcal{L}_{O,c}$, and an L2 loss on flow prediction $\mathcal{L}_F$. The total loss is:

$$\mathcal{L} = \alpha\mathcal{L}_{O,b} + \beta\mathcal{L}_{O,c} + \gamma\mathcal{L}_F. \tag{1}$$

4 Experiment

4.1 Dataset

We train, evaluate, and test our model on the Waymo open motion dataset (WOMD) based on the standard split and the filtered scenarios.

4.2 Metrics

We follow the standard challenge metrics, which include *AUC* and *Soft IoU* for observed occupancy, occluded occupancy, and flow-grounded occupancy, as well as the *end-point error (EPE)* that measures the error of flow prediction.

4.3 Model Details

Our model is illustrated in Fig 1. We use the standard VGG-16 model from *torchvision.models* as our rasterized encoder, and follow the implementations of VectorNet as in [5][3]. The input to the VectorNet includes i) a set of road element vectors with a shape of $B \times N_R \times 9$, where B is the batch size, $N_R = 10000$ is the maximum number of road element vectors, and the last dimension of 9 represents the positions (x, y) and headings $(\cos\theta, \sin\theta)$ of two end points in each vector and the vector id; ii) a set of agent vectors with a shape of $B \times 1280 \times 9$, including the vectors of up to 128 agents in a scene, where each agent has 10 vectors from the observed positions. We follow VectorNet by first running a local graph over each traffic element based on their ids and second running a global graph over all local features to obtain a vectorized feature with a shape of $B \times 128 \times N$, where N is the total number of traffic elements, including road elements and agents. We further quadrupled the size of the feature through an MLP layer to obtain a final vectorized feature V with a shape of $B \times 512 \times N$, so that its feature size is consistent with the channel size of the image feature, as discussed in the following paragraph.

[3]Code available at https://github.com/Tsinghua-MARS-Lab/DenseTNT

Model	Observed		Occluded		Flow-Grounded		
	AUC ↑	Soft IoU ↑	AUC ↑	Soft IoU ↑	AUC ↑	Soft IoU ↑	EPE ↓
HorizonOccFlowPred.	**0.803**	0.235	0.165	0.017	**0.839**	**0.633**	3.672
Look Around	0.801	0.234	0.139	0.029	0.825	0.549	**2.619**
Temporal Query	0.757	0.393	0.171	0.040	0.778	0.465	3.308
STrajNet	0.751	0.482	0.161	0.018	0.777	0.555	3.587
3D-STCNN	0.691	0.412	0.115	0.021	0.733	0.468	4.181
Motionnet	0.694	0.411	0.141	0.031	0.732	0.469	4.275
FTLS	0.618	0.318	0.085	0.019	0.689	0.431	9.612
OccFlowNet	0.667	0.391	0.111	0.026	0.678	0.443	6.636
VectorFlow	0.755	**0.488**	**0.174**	**0.045**	0.767	0.530	3.583

Table 1: Prediction performance on the test set. The best performed metrics are bolded and the grey cell indicates the ranking metric used by the WOMD benchmark. Our model achieves the best performance in three metrics.

Model	Observed		Occluded		
	AUC ↑	Soft IoU ↑	AUC ↑	Soft IoU ↑	EPE ↓
VectorFlow (VGG-only)	0.746	0.468	0.139	0.034	3.713
VectorFlow	**0.760**	**0.490**	**0.173**	**0.050**	**3.603**

Table 2: Prediction performance on the validation set. The best performed metrics are bolded. Using a vectorized representation helps boost the prediction performance.

We denote the output features of each VGG stage as $\{C_1, C_2, C_3, C_4, C_5\}$, and they have strides of $\{1, 2, 4, 8, 16\}$ pixels with respect to the input image and a hidden dimension of 512. The vectorized feature V is fused with the rasterized image feature C_5 with a shape of $B \times 512 \times 16 \times 16$ by a cross-attention module to obtain F_5 with the same shape. The query term of the cross-attention is the image feature C_5 flattened into $B \times 512 \times 256$ with 256 tokens, and the key and value term is the vectorized feature V with N tokens. We then concatenate F_5 and C_5 in the channel dimension and pass it through two 3×3 conv layers to obtain P_5 with a shape of $B \times 512 \times 16 \times 16$. P_5 is upsampled and concatenated with C_4 ($B \times 512 \times 32 \times 32$) by an FPN-style 2×2 upsampling module to generate U_4 with the same shape as C_4. Next, we perform another round of fusion between V and U_4 to obtain P_4 ($B \times 512 \times 32 \times 32$) following the same procedure, including cross attention. At the end, P_4 will be gradually upsampled by the FPN-style network and concatenated with $\{C_3, C_2, C_1\}$ to generate P_1 with a shape of $B \times 512 \times 256 \times 256$. We pass P_1 through two 3×3 conv layers to obtain the final output feature with a shape of $B \times 128 \times 256 \times 256$.

The decoder is a single 2D convolution layer with an input channel size of 128 and an output channel size of 32 (8 waypoints $\times$ 4 output dimensions).

4.4 Training Details

We train our model on the full training set of WOMD with a batch size of 32 for 16 epochs on 8 Nvidia A10 GPUs. We use an Adam optimizer and a learning rate scheduler that decays the learning rate by 50% every 5 epochs, with an initial value of 1e-3. The loss coefficients are $\alpha = \beta = 1000$, and $\gamma = 1$.

4.5 Results

We present the results of the challenge in Table 1. Our model achieves the best performance in three metrics, including both AUC and Soft IoU scores for occluded occupancy predictions.

Furthermore, we compare the performance of our model with a variant that only includes the VGG encoder. The results in Table 2 show that fusion helps improve the prediction performance, espe-

cially in occluded metrics. More specifically, the Occluded AUC and Occluded Soft IoU scores improve by 24.46% and 47.06%, respectively.

5 Conclusion

In this report, we present a simple but effective occupancy and flow predictor that combines two popular representations used in trajectory prediction. Our model achieves competitive performance compared to other challenge entries.

References

[1] R. Mahjourian, J. Kim, Y. Chai, M. Tan, B. Sapp, and D. Anguelov. Occupancy flow fields for motion forecasting in autonomous driving. *IEEE Robotics and Automation Letters*, 7(2): 5639–5646, 2022.

[2] S. Ettinger, S. Cheng, B. Caine, C. Liu, H. Zhao, S. Pradhan, Y. Chai, B. Sapp, C. R. Qi, Y. Zhou, et al. Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9710–9719, 2021.

[3] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[4] J. Gao, C. Sun, H. Zhao, Y. Shen, D. Anguelov, C. Li, and C. 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.

[5] J. Gu, C. Sun, and H. Zhao. DenseTNT: End-to-end trajectory prediction from dense goal sets. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15303–15312, 2021.