MapNeXt: Runner-up Solution of CVPR 2023 End-to-End Autonomous Driving Workshop

Toyota Li

Abstract

High-Definition (HD) maps are pivotal to autopilot navigation. Integrating the capability of lightweight HD map construction at runtime into a self-driving system recently emerges as a promising direction. In this surge, vision-only perception stands out, as a camera rig can still perceive the stereo information, let alone its appealing signature of portability and economy. The latest MapTR [5] architecture solves the online HD map construction task in an end-to-end fashion but its potential is yet to be explored. In this work, we present a full-scale upgrade of MapTR and propose MapNeXt, the next generation of HD map learning architecture, delivering major contributions from the model training and scaling perspectives. The proposed MapNeXt is runner-up in the online HD map construction competition of CVPR 2023 End-to-End Autonomous Driving workshop.

1. Approach

In this section, we first compile a short summary of the task and the most relevant literature. Next, we decompose the model architecture and discuss its components one by one.

1.1. Preliminary

Online HD map construction is an emerging topic in the autonomous driving community, which has not been widely investigated. Therefore, we decided to add a little background to provide easy access to a broad audience and ensure that the setup between different papers is consistent.

Map elements could have dynamic geometrical shape. For example, the lane divider is of open shape while the pedestrian crossing is of closed shape. The nature of varied shape makes it unavailable to model these map elements in a unified parametric manner. Thus, the open-shape and closed-shape map elements are approximated as polylines and polygons respectively, by sampling equidistant points on themselves. Formally, a map element can be discretized into an *ordered* set of points $V = [v_1, v_2, \cdots, v_{N_v}]$, where N_v is the total number of sampled points. This procedure is termed vectorization, so the processed HD map is called vectorized HD map accordingly.

VectorMapNet utilizes an auto-regressive model to sequentially emit the points $v_i, i = 1, 2, \cdots, N_v$ of one map element, which suffers from an inefficient inference. By contrast, MapTR relaxes the constraint of fixed order by expanding each single map element V to a set of permutation-equivalent map elements $\mathcal{V} = (V, \Gamma) = \{V^j = \gamma^j(V), j = 1, 2, \cdots, M\}$, where $\Gamma = \{\gamma^j, j = 1, 2, \cdots, M\}$ denotes a collection of M transformations that reorganize the ordered points of a map element but still reserve both its vectorized format and geometrical shape. In other words, $\forall V^i, V^j \in \mathcal{V}$, s.t., $i, j = 1, 2, \cdots, M, i \neq j$, then V^i and V^j are equivalent up to a permutation, described by $\gamma^j \circ (\gamma^i)^{-1}$.

After warming readers up, we shall step into more in-depth understanding and improved design of map element learning in the sequel.

1.2. Training

Following the encoder-decoder paradigm of DETR [1], to frame the map element prediction problem as sparse instance detection, a vast number of object-centric queries $Q = \{q_1, q_2, \cdots, q_N\}$ are created to cover all the map elements in one scene (N is greater than the maximal possible number of map elements). During inference, MapTR infers all the map elements in one shot and each map element is inferred from an individual query in the decoder. Specifically, each query $q_i \in \mathbb{R}^D, i = 1, 2, \cdots, N$ aggregates spatial information from the encoder feature via cross attention and outputs the corresponding classification and localization predictions $p_i^{(cls)} \in \mathbb{R}^C$ and $p_i^{(loc)} \in \mathbb{R}^2$, where C is the number of pre-defined categories of map elements. For brevity, we combine them into a whole prediction $p_i = [p_i^{(cls)}, p_i^{(loc)}] \in \mathbb{R}^{C+2}$ by concatenation.

Given a set of ground truths $G = \{g_1, g_2, \cdots, g_N\}$ that is padded with $\varnothing$ (no object) to the length of N, each ground truth g_i (actually a map element V_i in Section 1.1) is uniquely assigned to one query $q_{\pi(i)} \in Q$ as its label, where π is a permutation of the indices. The optimal bipartite matching $\hat{\pi}$ can be determined using the Hungarian algorithm [4] by minimizing the overall matching cost between all predictions and all ground truths. Once ground truth labels are defined for each query (as well as its corresponding prediction), the task-specific loss is derived as

$$\mathcal{L} = \sum_{i=1}^{N} \mathcal{L}_{\text{Hungarian}}(p_{\hat{\pi}(i)}, g_i), \tag{1}$$

where the Hungarian loss consists of classification, localization, and direction loss as MapTR. For ease of illustration, we only take into account the loss from the final Transformer decoder layer here, while in practice the total loss is summed up from all the decoder layers.

However, considering the limited number of map elements in a self-driving scenario, the bipartite matching strategy makes the neural network receive little ground truth supervision from few learnable queries. In consequence, the convergence speed and detection performance are reduced. Diving deep into MapTR, we notice that the permutation-equivalent transformation mentioned in Section 1.1 accidentally addresses this issue to some extent. Concretely speaking, since the queries are believed to be associated with spatial positions [1], by introducing numerous spatially-permuted versions of a map element, the originally one ground truth can be assigned to *distinct* queries now to create more supervised matching pairs. Thus, from our viewpoint, instead of "stabilizing the learning process" as claimed by MapTR, the major effect of modeling diverse permutation-equivalent map elements is essentially accelerating the convergence and improving the detection performance, as shown in Figure 5 and Table 2 of the original MapTR paper respectively.

Moreover, in the implementation of MapTR[1], *not all the possible* permutations of a map element are performed as stated in the paper. Instead, a *pre-defined* number of M permutations are applied for each map element. In other words, the applied permutations do not cover a full set, but merely a subset with a fixed cardinality M. For example, a polygon is only permuted with circular shifting but without reversing its direction in practice. Afterward, random samples are drawn if the total number of permuted samples exceeds the limit M. Now, ground truths are augmented as $G = \cup_{j=1}^{M} G^j = \cup_{j=1}^{M}\{g_1^j, g_2^j, \cdots, g_N^j\}$ with $g_i^j = \gamma^j(g_i)$ and the optimal assignment $\hat{\pi}(i,j)$ depends also on the index j. The overall loss exactly used by MapTR is calculated as

$$\mathcal{L} = \sum_{i=1}^{N}\sum_{j=1}^{M} \mathcal{L}_{\text{Hungarian}}(p_{\hat{\pi}(i,j)}, g_i^j). \tag{2}$$

Observed from the above equations, we further hypothesize that the top performance of MapTR roots in the increment of ground truths, but does not excessively demands a full coverage of *all possible* permuted ground truths. To back up this contradiction, we conduct a controlled experiment using MapTR-Tiny. The performance difference between a full and partial set of permutations is indeed marginal. In fact, the enhanced supervision furnished by duplicated ground truths is somewhat obscured by the highlighted permutation-invariant ground truth modeling, to which MapTR mostly attributes its success.

[1]https://github.com/hustvl/MapTR

Based on the understanding above, it is natural to further augment the supervision for neural networks from the standpoint of query. Mathematically, the original one set of queries is augmented to K sets as $Q = \cup_{k=1}^{K} Q^k = \cup_{k=1}^{K}\{q_1^k, q_2^k, \cdots, q_N^k\}$. Note that during bipartite matching, the ground truths G are still only matched with one set of queries at a time, for the purpose of avoiding post-processing. Therefore, the optimal assignment may be inconsistent across different sets of queries. For the k^{th} set of queries, it is represented as $\hat{\pi}_k(i,j)$. Equipped with the augmented queries, the loss is written as

$$\mathcal{L} = \sum_{k=1}^{K}\sum_{i=1}^{N}\sum_{j=1}^{M} \mathcal{L}_{\text{Hungarian}}(p^k_{\hat{\pi}_k(i,j)}, g_i^j). \tag{3}$$

Along different axes, augmenting the query could be realized in two modes, sequential or parallel. Resembling the style of DenseNet [3], the sequential mode reuses the queries coming from the previous Transformer decoder layers, so the number of queries will iteratively increase from shallower to deeper Transformer decoder layers. The parallel mode simply uses multiple sets of queries from the very start of Transformer decoder and the same number of queries exists in consecutive Transformer decoder layers. No matter under which mode, in every Transformer layer, the self-attention interaction only occurs within each individual set of queries. It is noteworthy that the augmented queries are merely used during training and we only apply one set of query Q^1 during inference, keeping the deployment latency untouched.

In terms of the sequential mode, iteratively accumulating the last two Transformer layers' queries has exhausted the GPU memory, but only improves the mAP marginally. So we insist on the parallel mode unless otherwise specified. Regarding the parallel mode, we explore the number of additional query set that spans a wide range from 0 to 10. We find that the performance consistently improves with the increasing query set and still does not saturate until running out of GPU memory. We choose to conservatively enlarge the number of query set to 1+10 by default.

In addition, the positional embedding fed to the decoder can be born in different formats. MapTR generates reference locations from implicitly initialized position embedding, which invokes an ambiguity in its geometric meaning, since such an embedding does not have any notion of spatial distribution prior. On the contrary, we advocate producing position embedding with explicitly initialized reference locations. Unlike MapTR that requires a linear projection layer, a sinusoidal encoding function [6] without trainable parameters is leveraged in our model to map a normalized 2D location into the latent embedding space. As a result, the inference process is even accelerated mildly. Thanks to the positional information straightforwardly injected into the learnable queries, our decoder is optimized in a more easy and interpretable manner.

1.3. Scaling

For the map element decoder, we find that sufficient decoder network capacity is necessary to digest more queries, so as to guarantee an improved performance. We decide on a balanced combination of 100 instance query and 2048d FFN. For the image encoder, we employ the ConvNeXt V2-Huge backbone [8] that is pre-trained on ImageNet-22K [2].

2. Experiment Details

The competition dataset is built on top of Argoverse2 [7], where the front-view image is portrait while the others are landscape, so pre-processing steps include resizing different views into a unified image resolution. The input images are resized with a scaling factor of 0.9. Except that, most of the data pre-processing steps follow the precedent practice of MapTR [5].

Note that the model is merely trained for 24 epochs and our submission is the second earliest on the public leaderboard, so there leaves much room for further enhancement.

References

[1] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 213–229, Cham, 2020. Springer International Publishing. 1, 2

[2] 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, 2009. 3

[3] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 2

[4] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, March 1955. 1

[5] Bencheng Liao, Shaoyu Chen, Xinggang Wang, Tianheng Cheng, Qian Zhang, Wenyu Liu, and Chang Huang. MapTR: Structured modeling and learning for online vectorized HD map construction. In *The Eleventh International Conference on Learning Representations*, 2023. 1, 3

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. 3

[7] Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, Deva Ramanan, Peter Carr, and James Hays. Argoverse 2: Next generation datasets for self-driving perception and forecasting. In J. Vanschoren and S. Yeung, editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1. Curran, 2021. 3

[8] Sanghyun Woo, Shoubhik Debnath, Ronghang Hu, Xinlei Chen, Zhuang Liu, In So Kweon, and Saining Xie. ConvNeXt V2: Co-designing and Scaling ConvNets with Masked Autoencoders. *arXiv e-prints*, page arXiv:2301.00808, Jan. 2023. 3