MapNeXt: Revisiting Training and Scaling Practices for Vectorized HD Map Construction

Toyota Li

toyota.li@foxmail.com

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 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. After shedding light on the training dynamics of MapTR and exploiting the supervision from map elements thoroughly, MapNeXt-Tiny raises the mAP of MapTR-Tiny from 49.0% to 54.8%, without any architectural modifications. Enjoying the fruit of map segmentation pre-training, MapNeXt-Base further lifts the mAP up to 63.9% that has already outperformed the prior art, a multi-modality MapTR, by 1.4% while being $\sim 1.8\times$ faster. Towards pushing the performance frontier to the next level, we draw two conclusions on practical model scaling: increased query favors a larger decoder network for adequate digestion; a large backbone steadily promotes the final accuracy without bells and whistles. Building upon these two rules of thumb, MapNeXt-Huge achieves state-of-the-art performance on the challenging nuScenes benchmark. Specifically, we push the mapless vision-only single-model performance to be over 78% for the first time, exceeding the best model from existing methods by 16%. This work was done before the beginning of 2023 and we were honored as the runner-up of track 2 in the VCAD CVPR 2023 Workshop's challenge.

Keywords: BEV perception · Online HD Map Construction · Autonomous Driving

1 Introduction

Autonomous driving is undoubtedly an attractive and challenging field nowadays, where perceiving the environment surrounding the ego-vehicle both accurately and holistically is a crucial pillar. Thus, High-Definition (HD) map with abundant geometric and semantic information is an indispensable ingredient for autopilot. Until recently, offline HD maps constructed with SLAM-based methods [52] remain the mainstay of the self-driving community. Though precise and reliable, the global HD maps not only call for massive labor for annotation but also become expensive to maintain once the real-world environment changes. Not surprisingly, it becomes a trend among the industrial enterprises to pile into "Mapless Driving". That is to say, the ever-increasing focus has been shifted towards building a local HD map on the fly with onboard sensory observations, providing great scalability and timeliness. The rationality of this choice is further backed by the fact that humans could already infer the surrounding scene geometry and semantics straightforwardly based on visual cues, without referring to a map.

Regarding vehicle-mounted sensors, LiDAR is adept at capturing geometry information in the scene, but its return lacks dense details and texture patterns compared to high-resolution cameras. In addition, from the perspective of industry, considering the bulkiness and costliness of LiDAR, mass production of LiDAR-equipped automobiles is hardly accessible. In contrast, the compactness and popularity of cameras render vision-only perception desiderata. Therefore in this work, we prefer to embrace the surrounding-view cameras as the only input source for perception on the vehicular platforms.

Some early attempts formulate online HD map construction as per-pixel prediction of a rasterized map. As a representative work, HDMapNet [21] dissects the entire problem into several sub-tasks, where the principal one is Bird's Eye View (BEV) semantic segmentation while the rest auxiliary tasks supplement instance information by means of complicated post-processing. Virtually, a map element is desired to be defined in a vectorized format, *i.e.*, an *ordered* set of points, that can be readily consumed by subsequent motion forecasting and planning modules. In this spirit, VectorMapNet [26] evolves the task in hand from dense pixel-level segmentation to sparse instance-level detection, of which the detection head features DETR [5] style, characterizing an end-to-end pipeline. However, each vectorized map element is deemed to be in a point sequence form and such an ordered output need to be arranged by an additional auto-regressive model, inevitably dragging down the inference speed. Although modeling online HD map construction as a sparse detection task is a milestone towards simplicity and efficiency, the remaining annoyance is how to elegantly cope with the order of output points. MapTR [23] stretches this line of research, giving birth to a completely streamlined architecture with parallelized output. Meanwhile, the determination of point order is sidestepped by creating a bunch of permutation-equivalent ground truths to match prediction outputs in diverse orders. The permutations are fulfilled by a collection of geometrical transformations, such as flipping and shifting, which could get rid of the matching ambiguity between many potential network predictions and one ground truth with the specific point order, during both model optimization and evaluation. Consequently, MapTR not only extends the intriguing property of end-to-end execution through to the entire neural network, but also demonstrates performance superiority over precedent models.

We meticulously study the MapTR architecture and suggest that there exist two concerns with respect to its performance. On the one hand, the reasons behind its compelling performance are under-explored. On the other hand, albeit with a breakthrough, the overall performance of MapTR is still unpleasant in real-world applications to date. For these two concerns, we mainly resolve them from the perspective of model *training* and *scaling*:

Training. We provide an in-depth analysis of the advanced mechanism of MapTR through the lens of training dynamics and reveal that its improved performance is inherently attributed to augmented ground truths. Unfortunately, this root cause is less publicized by the authors of MapTR. Since the bipartite matching policy from DETR marries a unique prediction among a large pool of (thousands of) queries to one map element, scarce supervision signal is propagated back to the neural network. Augmenting ground truths overcomes the drawbacks of sparse supervision, which is unintentionally materialized in MapTR by equivalent permutations of ground truths. On top of that, we disclose new chances of augmenting the query to permit map elements to be supervised more often, substantially ameliorating the performance. Besides our advance in the map element decoder, we also underscore the vital role of proper domain knowledge transfer during image encoder pre-training. Note that all the above findings are associated with model training schedules, that could plug and play for online HD map construction, without introducing any additional computation cost during inference.

Scaling. Foundation models lately make inroads into the vision domain and reigned supreme [29, 51]. For online HD map construction exceptionally, to our best knowledge, we are the first to scale up the model to probe the performance ceiling. Since efficiency is always the first-class citizen in the self-driving kingdom, we also keep in mind that the large-scale architecture design should be still friendly to parallel computing chips. For instance, we shall stick to pure convolution-based image encoders and expand the decoder through the dimension of network width.

To summarize, we present an omni-scale reloading of the MapTR architecture ranging from image encoder to map element decoder based on improved *training* and *scaling* techniques. The proposed architecture is fittingly dubbed as MapNeXt. Our core contributions in this work are threefold:

1. Targeted at onboard models, we propose improved training techniques including augmenting the query for map element decoder and preparing dedicated pre-training for image encoder, bringing about striking performance gains without adding any inference budget.
2. Targeted at offboard models, we offer golden guidelines on model scaling, such as matching the decoder capacity with the quantity of decoding query. We also unveil the feasibility of translating the rapid development of modern image backbones to online HD map learning for the first time.
3. On the competitive nuScenes benchmark, our real-time MapNeXt-Tiny improves over the strong MapTR baseline by 5% mAP or so, running even slightly faster; our non-real-time single-modal MapNeXt-Huge sets the new state-of-the-art with a 78.5 mAP, beating the known best multi-modal model by 16 mAP.

We anticipate that our MapNeXt could serve as a promising foundation, prompting more researchers to devote themselves to the arena of online vectorized HD map construction.

2 Related Work

2.1 Vision-only Obstacle Perception for Autopilot

Moving/static obstacle perception in autonomous driving is closely related to the techniques of 3D object detection in the field of computer vision. Only with images as the input signal, there mainly exist two branches of research, one is the bottom-up approach, the other is the top-down approach. The former one is represented by CaDDN [38] and BEVDet [16] which maintain the information flow inside the neural network naturally in a feed-forward fashion: extracting 2D image features, explicitly lifting them to the BEV space, and detecting objects in this space. During the process, they mostly employ well-developed existing components (image backbone [13, 24] and 3D object detection head [19, 50]), except the specialized PV-to-BEV view transformation module [37]. The latter one is represented by DETR3D [45] and PETR [27] which cast representative queries as the detected objects and then trace back to the corresponding image features. Specifically, they adopt object-centric queries to aggregate the latent features in the multi-view image space according to the camera intrinsics and extrinsics. Next, these queries are refined in a cascade style with a stack of Transformer layers for the final classification and localization tasks. In general, online HD map construction is resolved similarly to instance detection with BEV representation learning, but the details can be very different.

2.2 Vision-only Map Perception for Autopilot

Static element perception in autonomous driving partly refers to semantic map element prediction, a.k.a., online HD map construction. Borrowing the lessons from recent obstacle perception practices, map perception is also established in the BEV space using only vehicle-mounted camera sensors. The online HD Map construction problem is initially solved in a two-step paradigm. For example, the pioneering HDMapNet [21] predicts pixel-wise semantic categories, instance embedding, and direction simultaneously. Heuristic post-processing is necessitated to obtain structural information from the individual dense prediction results. Differently, VectorMapNet [26] proposes to organize the map elements in a vectorized form that is helpful to downstream tasks [11] and regard the problem as set prediction [5], while it still falls into a two-step solution where an auto-regressive generative model is involved. MapTR [23] first enables end-to-end HD map learning by supervising the hierarchical queries with a variety of permutation-equivalent ground truths. We appreciate the efficiency and impressive performance of MapTR [23], so we develop our method on the basis of it.

2.3 Vision Transformer

ViT [10] demonstrates that as a fundamental image encoder architecture, Transformer could deliver on par performance as full-fledged ConvNets [31]. DETR [5] and several follow-up works [22, 54] demonstrate that as a decoder, with the aid of bipartite matching, Transformer eliminates the necessity of NMS post-processing and thus lends unprecedented succinctness to cutting-edge detectors, without compromising their efficacy. In this work, we migrate DETR-style head to the field of online HD map construction as well, and analyze its behavior mainly through the lens of training dynamics. Part of this work is technically similar to Group DETR [6], but our analysis originates from the introspection of MapTR and we put forward different variants for query augmentation, in complementary to ground truth augmentation.

3 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.

3.1 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 $\mathcal{V} = (V, \mathit{\Gamma}) = \{V^j = \gamma^j(V), j = 1, 2, \cdots, M\}$, where $\mathit{\Gamma} = \{\gamma^j, j = 1, 2, \cdots, M\}$ 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, $\forall V^i, V^j \in \mathcal{V}, \text{s.t.}, i, j = 1, 2, \cdots, M, i \neq j$, 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.

3.2 Decoder

Training Following the encoder-decoder paradigm of DETR [5], 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 Sec. 3.1) 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 [18] 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 Sec. 3.1 accidentally addresses this issue to some extent. Concretely speaking, since the queries are believed to be associated with spatial positions [5], 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 Fig. 5 and Tab. 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 *predefined* number of M permutations are applied for each map element. In other

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

Table 1: Two different ground truth permutation strategies give rise to similar performance. Each map element is either permuted with all the possibilities (full set) or M randomly sampled transformations (sub set).

GT permutation	AP_{ped}	AP_{lane}	AP_{road}	mAP
full set (MapTR introduced)	43.6	51.2	52.3	49.0
sub set (MapTR implemented)	44.1	51.7	51.5	49.1

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. As shown in Tab. 1, 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.

As a corroboration, if MapTR chooses the unordered Chamfer distance cost for point-level matching in the Hungarian algorithm, one query would have the same distance to all the permuted versions of a map element, then the term of localization cost would completely lose its effect of discriminating different permuted ground truths. The merit of augmented localization supervision would be thereby attenuated. Table 8 of the MapTR paper shows that under such a situation, the final performance is lower than permuted ground truths combined with an ordered point-to-point localization cost but still higher than fixed-order ground truths (at least the augmented classification supervision still helps).

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

Table 2: Augmenting queries under two modes, where the parallel mode is more scalable and effective.

mode	#set of query	AP_{ped}	AP_{lane}	AP_{road}	mAP
sequential	cumulative	43.0	53.2	55.0	50.4
parallel	1	43.6	51.2	52.3	49.0
	1+5	49.9	54.2	56.9	53.7
	1+10	51.9	54.6	55.5	54.0
	1+20	52.7	55.8	55.9	54.8

Table 3: Comparison among different options of position embedding. Models are equipped with 1+10 sets of parallel queries as in Tab. 2, the same in Tab. 4.

position embedding	AP_{ped}	AP_{lane}	AP_{road}	mAP
emb → pos	51.9	54.6	55.5	54.0
pos → emb (sine)	49.9	57.3	57.8	55.0
pos → emb (linear)	50.7	56.6	58.0	55.1

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 [14], 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.

The results are summarized in Tab. 2. In terms of the sequential mode, iteratively accumulating the last two Transformer layers' queries has exhausted the GPU memory, but only improves the mAP by around 1%. 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 20. We find that the performance consistently improves with the increasing query set and still does not saturate until running out of GPU memory. The performance advantage of 20 additional sets of query comes at the cost of an enormous memory footprint, impeding our further exploration in the following, so we choose to conservatively enlarge the number of query set to 1+10 by default. Nevertheless, the ever-growing performance in Tab. 2 speaks for the potential of our method to be further unleashed. We expect to witness another performance leap as soon as larger memory is affordable.

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

Table 4: Interplay between the number of query and the dimension of FFN. The number of query refers to that of instance query in *a single query set*. All model variants are trained for 110 epochs to full convergence. The specification of timing is in Tab. 6.

#query	FFN dim.	#params.	FPS	AP_{ped}	AP_{lane}	AP_{road}	mAP
50	512	82.1M	16.5	66.2	71.7	73.0	70.3
75	512	82.1M	16.3	66.0	72.0	73.4	70.5
50	1024	83.9M	16.3	67.8	72.5	73.0	71.1
50	2048	87.6M	16.3	67.1	72.4	73.5	71.0
75	1024	84.0M	16.4	68.0	73.1	74.0	71.7

Table 5: MapTR-Tiny with a wide spectrum of backbones and pre-training tasks. $^\flat$ indicates neither the shallow network stages nor BN layers are frozen.

backbone	pretrain	AP_{ped}	AP_{lane}	AP_{road}	mAP
R18	ImageNet cls$^\flat$	39.9	49.4	48.7	46.0
	CurveLanes	40.9	49.7	49.0	46.5
R50	ImageNet cls$^\flat$	44.9	52.2	53.8	50.3
	nuImages det	40.4	49.3	52.0	47.3
V99	ImageNet cls$^\flat$	54.0	62.9	61.0	59.3
	ImageNet cls	52.0	61.1	60.5	57.9
	nuScenes det	56.3	64.9	65.2	62.1
	nuScenes seg$^\flat$	58.5	65.5	67.8	63.9
	nuScenes seg	58.4	66.9	66.5	63.9

sinusoidal encoding function [41] 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, from 20.0 to 20.3 FPS on an NVIDIA A100 GPU. We also consider a linear projection layer as an alternative for mapping, but it brings negligible gain. The comparison results are displayed in Tab. 3. Thanks to the positional information straightforwardly injected into the learnable queries, our decoder is optimized in a more easy and interpretable manner. To the best of our knowledge, the most related work might be Anchor DETR [44], but our design is simple yet effective in comparison to Anchor DETR: our position encoding is not only exclusive of multiple patterns of anchors, but also leads to a decent performance gain even forgoing neural network layers.

Scaling To scale up, sufficient decoder network capacity is necessary to digest more queries, so as to guarantee an improved performance. For this set of experiments, we use a stronger VoVNetV2 backbone [20], of which the elaboration is deferred to Sec. 3.3. Either trivially increasing the query number or widening the Feed-Forward Network (FFN) independently achieves limited improvement. In stark contrast, combing the two finally yields a high-performing model, as shown in Tab. 4. The listed results are non-trivial from the following aspects: *I.* fixing the FFN dimension to 512, increasing the query brings little gain (line 1&2), *II.* after widening the FFN dimension to 1024, the same increment of query (50 → 75) brings a larger gain (line 3&5 vs. line 1&2), *III.* fixing the number of query to 50, even widening the FFN dimension to 2048 achieves inferior performance to a balanced combination of 75 instance query and 1024d FFN, albeit with a 3.6M more parameter count (line 4&5).

3.3 Encoder

Training MapTR initializes their ResNet [13] or Swin Transformer [30] backbone network with ImageNet [9] pre-trained weights, which is an old-fashioned scheme for transfer learning. We perform a pilot study by initializing the ResNet-18 weights pre-trained on CurveLanes [48] with CondLaneNet [25], which slightly strengthens its final performance, as exhibited in Tab. 5. Still, one thing worth noting is that *not all kinds of* pre-training helps. The pre-training domain should be as close to our task of interest as possible, *i.e.*, map element learning, in favor of a successful transfer.

To complement this posit, we provide counter examples in Tab. 5. First of all, we adopt a ResNet-50 pre-trained on nuImages with Cascade R-CNN [4] as the backbone. This pre-training setting is said to boost up the camera-based BEV 3D object detection significantly [47], but it deteriorates the map element construction oppositely, possibly due to the domain gap between the pre-training objectives and the target task. Beyond that, we employ VoVNetV2-99 [20] as the backbone network that is also pre-trained on ImageNet. We find that a common routine of frozen Batch Normalization (BN) statistics [17] induces a performance drop of over 1% mAP. The phenomenon implies that ImageNet might also not be a proper source dataset for our pre-training, since most object-centric images in ImageNet deviate from the driving scenarios. Another potential reason is that ImageNet pre-trained network concentrates on the classification task, then it would take much effort to adapt the weights for other downstream tasks, so it helps less if the target task is more sensitive to localization [12].

All the above analyses taken into account, it is encouraged to pick a highly relevant task and dataset for backbone pre-training, in order to narrow down the domain gap. VoVNetV2-99, initialized with the weights that are successively trained on DDAD-15M with DD3D [35] and nuScenes with FCOS3D [42], is the *de facto* standard backbone of top-performing 3D object detectors on the nuScenes leaderboard. MapTR armed with the same bespoke weights outstrips its ImageNet pre-trained counterpart by a remarkable 4.2% mAP. To take one step further, we pre-train the VoVNetV2 backbone on a nuScenes BEV map segmentation task with PETRv2 [28], to enjoy the benefit of rich semantic features. As expected, another 1.8% higher mAP is reached powered by this more relevant pre-training task. By the way, dissimilar to the case in the last paragraph, even frozen BN statistics would not impair the performance of this model, thanks to the same nuScenes data statistics.

Scaling Scaling up the model for online vectorized HD map construction is rarely well-studied. To bridge this gap, we provide a family of MapNeXt variants from on-board to off-board architectures. It should be emphasized that model scaling is never as effortless as expected. For example, a ConvNeXt-XL [31] backbone pre-trained on ImageNet performs merely on par with or even inferior to a VoVNetV2-99 pre-trained on DDAD-15M for 3D object detection/map segmentation, despite with a 3.5× parameter amount[2]. Unlike the above plateau, we reveal that the online HD map construction task fortunately enjoys the profit

[2] Under the PETRv2 framework for BEV segmentation, ConvNeXt-XL obtains an IoU of 85.6%/47.6%/42.5% for the drive/lane/vehicle class, not as good as the result of 85.6%/48.9%/46.4% achieved by VoVNetV2-99.

of large-scale image backbone in Tab. 6, where there still exists no evidence of performance saturation with hundreds of parameters.

3.4 Neck

Last but not least, the neck commonly bridges the encoder and decoder part in a detection system, which is instantiated as a PV-to-BEV transformation module in the BEV-oriented detectors. The original publication of MapTR has already sweepingly explored this component, ranging from the classical Inverse Perspective Mapping (IPM) [33], to modern Lift-Splat [37], deformable attention [22] and Geometry-guided Kernel Transformer (GKT) [7]. We also do not find much difference among these variants during reproduction and simply inherit the GKT module chosen by MapTR.

4 Main Experiments

4.1 Dataset

nuScenes [2] is a widely-adopted benchmark for versatile autonomous driving tasks. It contains 1, 000 scenes of 20 seconds duration each, of which the key frames are annotated at 2Hz. The entire dataset is split into 700, 150, and 150 scenes for training, validation, and testing respectively. Each sample contains RGB images from 6 surrounding-view cameras, covering a horizontal FOV of 360°.

Following the convention [21, 23, 26], the categories of interest are pedestrian crossing, lane divider and road boundary. Similar to regular object detection tasks, the evaluation metric is also Average Precision (AP). The difference lies in that the distance between two map elements is measured with Chamfer distance (unlike the Intersection over Union between two bounding boxes). For a fair comparison to peer works [21, 23, 26], the distance thresholds are ranging from 0.5 to 1.5 with an interval of 0.5.

4.2 Implementation Details

In general, we primarily follow MapTR's training protocol. All model architectures are implemented with the PyTorch library [36] and the training is distributed on 8 NVIDIA A100 GPU devices with Automated Mixed Precision (AMP) [34]. The mini-batch size per device is set to 4, except for ConvNeXt-XL and InternImage-H which is halved. The training period lasts for 24 epochs for fast prototyping and 110 epochs for system-level comparison. The initial learning rate is 0.006 and is decayed following a half-cosine-shaped function. The learning rate of the backbone is multiplied by a factor of $^1/_{10}$ because it has been pre-trained. We find the final result is fairly robust to the initial learning rate, probably thanks to the AdamW optimizer [32]. The weight decay is fixed as 0.01 and the ℓ_2 norm of gradients is clipped to be no more than 35. The probability

Table 6: State-of-the-art comparison on the nuScenes `val` set. "C" and "L" stand for input modality of camera and LiDAR. "R18/50", "V99" and "EffNet" denote ResNet-18/50 [13], VoVNetV2-99 [20] and EfficientNet [40] respectively. "PP" is short for PointPillars [19]. All the latency is measured with a batch size of 1 after warmup. The entries in gray are reported by MapTR [23] while others are timed by ourselves. † indicates using 75 instance queries and 1024 FFN dimensions, while ‡ indicates using 80 instance queries and 2048 FFN dimensions.

Architecture	Modality	Backbone	Epochs	AP				#Param.	FPS	
				ped.	lane	road	avg.		RTX3090	A100
HDMapNet [21]	C	EffNet-B0	30	14.4	21.7	33.0	23.0	-	0.8	-
HDMapNet [21]	L	PP	30	10.4	24.1	37.9	24.1	-	1.0	-
HDMapNet [21]	C & L	EffNet-B0 & PP	30	16.3	29.6	46.7	31.0	-	0.5	-
VectorMapNet [26]	C	R50	110	36.1	47.3	39.3	40.9	-	2.9	-
VectorMapNet [26]	L	PP	110	25.7	37.6	38.6	34.0	-	-	-
VectorMapNet [26]	C & L	R50 & PP	110	37.6	50.5	47.5	45.2	-	-	-
MapTR-Nano [23]	C	R18	110	39.6	49.9	48.2	45.9	15.3M	29.2	50.5
MapTR-Tiny [23]	C	R50	24	46.3	51.5	53.1	50.3	35.9M	12.6	20.0
MapTR-Tiny [23]	C	R50	110	56.2	59.8	60.1	58.7	35.9M	12.6	20.0
MapTR-Tiny [23]	C	Swin-Tiny	24	45.2	52.7	52.3	50.1	39.9M	9.1	-
MapTR-Small [23]	C	Swin-Small	24	50.2	55.4	57.3	54.3	61.2M	7.3	-
MapTR-Base [23]	C	Swin-Base	24	50.6	58.7	58.4	55.9	99.2M	6.1	-
MapTR-Tiny [23]	L	SECOND	24	48.5	53.7	64.7	55.6	-	7.2	-
MapTR-Tiny [23]	C & L	R50 & SECOND	24	55.9	62.3	69.3	62.5	39.8M	5.2	6.2
MapNeXt-Tiny	C	R50	24	50.3	58.8	58.7	56.0	36.0M	12.7	20.3
MapNeXt-Tiny	C	R50	110	57.7	65.3	65.8	63.0	36.0M	12.7	20.3
MapNeXt-Base	C	V99	24	58.5	65.5	67.8	63.9	82.1M	9.3	16.5
MapNeXt-Base	C	V99	110	66.2	71.7	73.0	70.3	82.1M	9.3	16.5
MapNeXt-Base†	C	V99	110	67.8	73.1	74.1	71.7	84.0M	9.2	16.4
MapNeXt-Large	C	ConvNeXt-XL	24	66.7	72.4	70.4	69.8	360.9M	3.5	5.9
MapNeXt-Large	C	ConvNeXt-XL	110	71.5	74.9	74.7	73.7	360.9M	3.5	5.9
MapNeXt-Large‡	C	ConvNeXt-XL	110	73.7	78.4	76.2	76.1	366.5M	3.5	5.9
MapNeXt-Huge‡	C	InternImage-H	110	77.4	79.3	78.8	78.5	-	-	-

of stochastic depth [15] applied to ConvNeXt-XL is 0.4. The layer-wise learning rate decay [1] applied to InternImage-H is 0.95. Such regularization is important for the performance of Transformer-like models based on both previous experience [39] and our observations. Data pre-processing is consistent with the precedent practice [23], in order to isolate the role of model improvement from other compounding factors. The perception range is [-15.0m, 15.0m] along the X-axis and [-30.0m, 30.0m] along the Y-axis with reference to the ego-vehicle.

4.3 Quantitative Results

The model profiling and performance comparison are showcased in Tab. 6. MapNeXt-Tiny outperforms the MapTR-Tiny counterpart by a considerable gain of 5.7% mAP after training for 24 epochs, with marginally higher throughput. This performance gain holds as a 4.3% mAP when they are both trained longer until 110 epochs. MapNeXt-Tiny even surpasses MapTR-Base that utilizes Swin Transformer-base [25] as the backbone, with only 36% parameters and 48% latency. The results prove the effectiveness of our proposed model training strategies. Note that we follow MapTR's training recipe and *do not* tune the hyper-parameters optimized for MapTR, which may even put our MapNeXt at a *disadvantage* in comparison to MapTR.

To verify the generality of MapNeXt, we further replace the image backbone with VoVNetV2-99. The preferable map segmentation pre-training strategy in Tab. 5 is applied. The resulting MapNeXt-Base outperforms our own MapNeXt-Tiny by more than 7% mAP, preserving 81% of the throughput. When comparing MapNeXt-Base against the best-in-class vision-only MapTR within 24 epochs, it surpasses MapTR-Base in terms of both efficacy and efficiency, *i.e*, 8% higher mAP and 3.2 more FPS. In addition, we would like to figure out if vision-only MapNeXt-Base could rival the strongest MapTR model, a multi-modality MapTR-Tiny, that takes the best from both worlds of camera and LiDAR. Impressively, the answer is affirmative. MapNeXt-Base outperforms multi-modal MapTR-tiny by an 1.4% mAP while being 1.8 times fast. The overwhelming performance of MapNeXt-Base justifies the privilege of VoVNet pre-training on nuScenes map segmentation. Moreover, the multi-modal MapTR model harnesses sparse convolution [49] to process the input LiDAR point cloud, posing challenges to onboard deployment, while our MapNeXt is free of such complex operators.

To scale up, we elevate the model capacity to a magnitude of hundreds of parameters, by upgrading the image encoder to ConvNeXt-XL and InternImage-H. We observe that the corresponding MapNeXt-Large does not level off in performance but widens the gap between itself and multi-modal MapTR to 7.3% mAP. Echoed with the findings from Tab. 4, we increase the query number and FFN dimension together when scaling up the decoder, leading to a non-trivial 2.4% mAP increment without compromising the inference efficiency. MapNeXt-Large is not a real-time model but can be capitalized on for offboard settings, such as auto-labeling. Importantly, when compared with non-real-time VectorMapNet with only camera input, MapNeXt-Large nearly doubles the accuracy while still running 1.2× faster. It is also worth mentioning that an over-fitting phenomenon has been observed in the later period of training, for which the limited input image resolution (0.5× resizing) is blamed. Though, to keep a clear comparison of model architecture, we would rather not increase the input scaling factor here but substantiate its effectiveness when participating in the CVPR 2023 challenge afterward in Sec. 5. Finally, armed with the InternImage-H [43] backbone network that is pre-trained on COCO-Stuff [3] and ADE20K [53] using Mask2Former [8], the online HD map construction performance of MapNeXt-Huge is lifted to an unparalleled 78.5% mAP.

5 Challenge Results

Built upon the skeleton of MapNeXt, our entry into the online HD map construction competition of CVPR 2023 Vision-Centric Autonomous Driving (VCAD)

Workshop and Joint CVPR 2023 End-to-End Autonomous Driving workshop wins the honorable runner-up with a 73.65 mAP on the test server *without any test time augmentation or model ensemble tricks*, outperforming the official baseline method by a notable 31.4 mAP. Note that the model is merely trained for 24 epochs and our submission is the second earliest one (∼20 days before deadline) on the public leaderboard, so there leaves much room for further enhancement. The dataset of this challenge is set up via reshaping Argoverse2 [46], where the front-view image is portrait while the others are landscape, so pre-processing steps include resizing the front view into a unified image resolution 1550×2048. Then, the input images are resized with a scaling factor of 0.9 to avoid overfitting. Excepth that, most of the data pre-processing steps follow the precedent practice of MapTR [23]. During this challenge, the general applicability and potential representation ability of MapNeXt are proved again in a different benchmark.

6 Conclusion

In this work, we rethink the optimization hindrance of MapTR and reinforce the training strategies with enriched informative queries. We additionally pinpoint the critical importance of appropriate pre-training. Since the self-driving world prioritizes efficiency, we highlight that these improved training techniques could be inserted into the existing architecture tailored for online HD map construction without interfering in the inference. Besides lightweight onboard models, we also build offboard ones with empirical scaling principles. In summary, we construct the model family named MapNeXt with a wide coverage of onboard and offboard architectures, attaining leading performance while retaining high throughput. Particularly, MapNeXt-Large/Huge sets a new state-of-the-art on the online HD map construction track of the public nuScenes benchmark. We hope this work paves a path to real-world applications of online vectorized HD map construction in autonomous driving.

References

1. Bao, H., Dong, L., Piao, S., Wei, F.: BEit: BERT pre-training of image transformers. In: International Conference on Learning Representations (2022), https://openreview.net/forum?id=p-BhZSz59o4
2. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuscenes: A multimodal dataset for autonomous driving. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2020)
3. Caesar, H., Uijlings, J., Ferrari, V.: Coco-stuff: Thing and stuff classes in context. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2018)
4. Cai, Z., Vasconcelos, N.: Cascade r-cnn: Delving into high quality object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2018)
5. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.M. (eds.) Computer Vision – ECCV 2020. pp. 213–229. Springer International Publishing, Cham (2020)
6. Chen, Q., Chen, X., Wang, J., Zhang, S., Yao, K., Feng, H., Han, J., Ding, E., Zeng, G., Wang, J.: Group detr: Fast detr training with group-wise one-to-many assignment. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 6633–6642 (October 2023)
7. Chen, S., Cheng, T., Wang, X., Meng, W., Zhang, Q., Liu, W.: Efficient and Robust 2D-to-BEV Representation Learning via Geometry-guided Kernel Transformer. arXiv e-prints arXiv:2206.04584 (Jun 2022). https://doi.org/10.48550/arXiv.2206.04584
8. Cheng, B., Misra, I., Schwing, A.G., Kirillov, A., Girdhar, R.: Masked-attention mask transformer for universal image segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1290–1299 (June 2022)
9. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition. pp. 248–255 (2009). https://doi.org/10.1109/CVPR.2009.5206848
10. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: International Conference on Learning Representations (2021), https://openreview.net/forum?id=YicbFdNTTy
11. Gao, J., Sun, C., Zhao, H., Shen, Y., Anguelov, D., Li, C., Schmid, C.: Vectornet: Encoding hd maps and agent dynamics from vectorized representation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2020)
12. He, K., Girshick, R., Dollar, P.: Rethinking imagenet pre-training. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (October 2019)
13. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2016)

14. Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (July 2017)
15. Huang, G., Sun, Y., Liu, Z., Sedra, D., Weinberger, K.Q.: Deep networks with stochastic depth. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) Computer Vision – ECCV 2016. pp. 646–661. Springer International Publishing, Cham (2016)
16. Huang, J., Huang, G., Zhu, Z., Ye, Y., Du, D.: BEVDet: High-performance Multi-camera 3D Object Detection in Bird-Eye-View. arXiv e-prints arXiv:2112.11790 (Dec 2021). https://doi.org/10.48550/arXiv.2112.11790
17. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Bach, F., Blei, D. (eds.) Proceedings of the 32nd International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 37, pp. 448–456. PMLR, Lille, France (07–09 Jul 2015), https://proceedings.mlr.press/v37/ioffe15.html
18. Kuhn, H.W.: The Hungarian method for the assignment problem. Naval Research Logistics Quarterly **2**(1-2), 83–97 (March 1955). https://doi.org/10.1002/nav.3800020109, https://ideas.repec.org/a/wly/navlog/v2y1955i1-2p83-97.html
19. Lang, A.H., Vora, S., Caesar, H., Zhou, L., Yang, J., Beijbom, O.: Pointpillars: Fast encoders for object detection from point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)
20. Lee, Y., Park, J.: Centermask: Real-time anchor-free instance segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2020)
21. Li, Q., Wang, Y., Wang, Y., Zhao, H.: Hdmapnet: An online hd map construction and evaluation framework. In: 2022 International Conference on Robotics and Automation (ICRA). pp. 4628–4634 (2022). https://doi.org/10.1109/ICRA46639.2022.9812383
22. Li, Z., Wang, W., Li, H., Xie, E., Sima, C., Lu, T., Qiao, Y., Dai, J.: Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. In: Avidan, S., Brostow, G., Cissé, M., Farinella, G.M., Hassner, T. (eds.) Computer Vision – ECCV 2022. pp. 1–18. Springer Nature Switzerland, Cham (2022)
23. Liao, B., Chen, S., Wang, X., Cheng, T., Zhang, Q., Liu, W., Huang, C.: MapTR: Structured modeling and learning for online vectorized HD map construction. In: The Eleventh International Conference on Learning Representations (2023), https://openreview.net/forum?id=k7p_YAO7yE
24. Lin, T.Y., Dollar, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (July 2017)
25. Liu, L., Chen, X., Zhu, S., Tan, P.: Condlanenet: A top-to-down lane detection framework based on conditional convolution. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 3773–3782 (October 2021)
26. Liu, Y., Yuan, T., Wang, Y., Wang, Y., Zhao, H.: VectorMapNet: End-to-end vectorized HD map learning. In: Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., Scarlett, J. (eds.) Proceedings of the 40th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 202, pp. 22352–22369. PMLR (23–29 Jul 2023), https://proceedings.mlr.press/v202/liu23ax.html

27. Liu, Y., Wang, T., Zhang, X., Sun, J.: Petr: Position embedding transformation for multi-view 3d object detection. In: Avidan, S., Brostow, G., Cissé, M., Farinella, G.M., Hassner, T. (eds.) Computer Vision – ECCV 2022. pp. 531–548. Springer Nature Switzerland, Cham (2022)
28. Liu, Y., Yan, J., Jia, F., Li, S., Gao, A., Wang, T., Zhang, X.: Petrv2: A unified framework for 3d perception from multi-camera images. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 3262–3272 (October 2023)
29. Liu, Z., Hu, H., Lin, Y., Yao, Z., Xie, Z., Wei, Y., Ning, J., Cao, Y., Zhang, Z., Dong, L., Wei, F., Guo, B.: Swin transformer v2: Scaling up capacity and resolution. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 12009–12019 (June 2022)
30. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 10012–10022 (October 2021)
31. Liu, Z., Mao, H., Wu, C.Y., Feichtenhofer, C., Darrell, T., Xie, S.: A convnet for the 2020s. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 11976–11986 (June 2022)
32. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: International Conference on Learning Representations (2019), https://openreview.net/forum?id=Bkg6RiCqY7
33. Mallot, H.A., Bülthoff, H.H., Little, J.J., Bohrer, S.: Inverse perspective mapping simplifies optical flow computation and obstacle detection. Biol. Cybern. **64**(3), 177–185 (jan 1991). https://doi.org/10.1007/BF00201978, https://doi.org/10.1007/BF00201978
34. Micikevicius, P., Narang, S., Alben, J., Diamos, G., Elsen, E., Garcia, D., Ginsburg, B., Houston, M., Kuchaiev, O., Venkatesh, G., Wu, H.: Mixed precision training. In: International Conference on Learning Representations (2018), https://openreview.net/forum?id=r1gs9JgRZ
35. Park, D., Ambrus, R., Guizilini, V., Li, J., Gaidon, A.: Is pseudo-lidar needed for monocular 3d object detection? In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 3142–3152 (October 2021)
36. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 32. Curran Associates, Inc. (2019), https://proceedings.neurips.cc/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf
37. Philion, J., Fidler, S.: Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.M. (eds.) Computer Vision – ECCV 2020. pp. 194–210. Springer International Publishing, Cham (2020)
38. Reading, C., Harakeh, A., Chae, J., Waslander, S.L.: Categorical depth distribution network for monocular 3d object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 8555–8564 (June 2021)

39. Steiner, A.P., Kolesnikov, A., Zhai, X., Wightman, R., Uszkoreit, J., Beyer, L.: How to train your vit? data, augmentation, and regularization in vision transformers. Transactions on Machine Learning Research (2022), https://openreview.net/forum?id=4nPswr1KcP
40. Tan, M., Le, Q.: EfficientNet: Rethinking model scaling for convolutional neural networks. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 97, pp. 6105–6114. PMLR (09–15 Jun 2019), https://proceedings.mlr.press/v97/tan19a.html
41. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 30. Curran Associates, Inc. (2017), https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf
42. Wang, T., Zhu, X., Pang, J., Lin, D.: Fcos3d: Fully convolutional one-stage monocular 3d object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops. pp. 913–922 (October 2021)
43. Wang, W., Dai, J., Chen, Z., Huang, Z., Li, Z., Zhu, X., Hu, X., Lu, T., Lu, L., Li, H., Wang, X., Qiao, Y.: Internimage: Exploring large-scale vision foundation models with deformable convolutions. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 14408–14419 (June 2023)
44. Wang, Y., Zhang, X., Yang, T., Sun, J.: Anchor detr: Query design for transformer-based detector. Proceedings of the AAAI Conference on Artificial Intelligence **36**(3), 2567–2575 (Jun 2022). https://doi.org/10.1609/aaai.v36i3.20158, https://ojs.aaai.org/index.php/AAAI/article/view/20158
45. Wang, Y., Guizilini, V.C., Zhang, T., Wang, Y., Zhao, H., Solomon, J.: Detr3d: 3d object detection from multi-view images via 3d-to-2d queries. In: Faust, A., Hsu, D., Neumann, G. (eds.) Proceedings of the 5th Conference on Robot Learning. Proceedings of Machine Learning Research, vol. 164, pp. 180–191. PMLR (08–11 Nov 2022), https://proceedings.mlr.press/v164/wang22b.html
46. Wilson, B., Qi, W., Agarwal, T., Lambert, J., Singh, J., Khandelwal, S., Pan, B., Kumar, R., Hartnett, A., Kaesemodel Pontes, J., Ramanan, D., Carr, P., Hays, J.: Argoverse 2: Next generation datasets for self-driving perception and forecasting. In: Vanschoren, J., Yeung, S. (eds.) Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks. vol. 1. Curran (2021), https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/file/4734ba6f3de83d861c3176a6273cac6d-Paper-round2.pdf
47. Xie, E., Yu, Z., Zhou, D., Philion, J., Anandkumar, A., Fidler, S., Luo, P., Alvarez, J.M.: M^{2}BEV: Multi-Camera Joint 3D Detection and Segmentation with Unified Birds-Eye View Representation. arXiv e-prints arXiv:2204.05088 (Apr 2022). https://doi.org/10.48550/arXiv.2204.05088
48. Xu, H., Wang, S., Cai, X., Zhang, W., Liang, X., Li, Z.: Curvelane-nas: Unifying lane-sensitive architecture search and adaptive point blending. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.M. (eds.) Computer Vision – ECCV 2020. pp. 689–704. Springer International Publishing, Cham (2020)
49. Yan, Y., Mao, Y., Li, B.: Second: Sparsely embedded convolutional detection. Sensors **18**(10) (2018). https://doi.org/10.3390/s18103337, https://www.mdpi.com/1424-8220/18/10/3337

50. Yin, T., Zhou, X., Krahenbuhl, P.: Center-based 3d object detection and tracking. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 11784–11793 (June 2021)
51. Zhai, X., Kolesnikov, A., Houlsby, N., Beyer, L.: Scaling vision transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 12104–12113 (June 2022)
52. Zhang, J., Singh, S.: LOAM: lidar odometry and mapping in real-time. In: Robotics: Science and Systems X, University of California (2014)
53. Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., Torralba, A.: Scene parsing through ade20k dataset. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (July 2017)
54. Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J.: Deformable DETR: Deformable transformers for end-to-end object detection. In: International Conference on Learning Representations (2021), https://openreview.net/forum?id=gZ9hCDWe6ke