Redundant Queries in DETR-Based 3D Detection Methods: Unnecessary and Prunable

Lizhen Xu[1*], Zehao Wu[1*], Wenzhao Qiu[1], Shanmin Pang[1†], Xiuxiu Bai[1], Kuizhi Mei[2], Jianru Xue[2]

[1]School of Software Engineering, Xi'an Jiaotong University
[2]School of Artificial Intelligence, Xi'an Jiaotong University
{Corona@stu., 3124358020Wu@stu., a2801551754@stu., pangsm@, xiubai@, meikuizhi@, jrxue@}xjtu.edu.cn

Abstract

Query-based models are extensively used in 3D object detection tasks, with a wide range of pre-trained checkpoints readily available online. However, despite their popularity, these models often require an excessive number of object queries, far surpassing the actual number of objects to detect. The redundant queries result in unnecessary computational and memory costs. In this paper, we find that not all queries contribute equally – a significant portion of queries have a much smaller impact compared to others. Based on this observation, we propose an embarrassingly simple approach called **G**radually **P**runing **Q**ueries (GPQ), which prunes queries incrementally based on their classification scores. A key advantage of GPQ is that it requires no additional learnable parameters. It is straightforward to implement in any query-based method, as it can be seamlessly integrated as a fine-tuning step using an existing checkpoint after training. With GPQ, users can easily generate multiple models with fewer queries, starting from a checkpoint with an excessive number of queries. Experiments on various advanced 3D detectors show that GPQ effectively reduces redundant queries while maintaining performance. Using our method, model inference on desktop GPUs can be accelerated by up to 1.35x. Moreover, after deployment on edge devices, it achieves up to a 67.86% reduction in FLOPs and a 65.16% decrease in inference time.

Code — https://github.com/iseri27/Gpq
Extended version — https://arxiv.org/pdf/2412.02054

1 Introduction

Detection of 3D objects is crucial for autonomous driving (Ma et al. 2024; Fan et al. 2023; Wang et al. 2025). Among various algorithms, DETR-based methods (Carion et al. 2020; Wang et al. 2022; Liu et al. 2022; Wang, Jiang, and Li 2023; Wang et al. 2023) stand out for their end-to-end detection capabilities without relying on hand-crafted components. A core element of these models is the use of pre-defined queries, derived from reference points (Zhu et al. 2020; Fang et al. 2024; Liu et al. 2023a; Wang, Jiang, and Li 2023; Wang et al. 2023; Jiang et al. 2024; Li et al. 2022; Yang et al. 2023; Liao et al. 2023a,b). These queries are first refined via self-attention, then interact with image features through cross-attention. The resulting queries are passed to MLPs to predict classification scores and 3D bounding boxes.

Queries						
# Ref	# Pro.	# Tot.	mAP↑	NDS↑	Mem.(MiB)↓	FPS↑
1288	512	1800	39.62%	0.4965	2580	12.7
1074	426	1500	39.37%	0.4945	2520	15.7
858	342	1200	39.23%	0.4960	2466	15.8
644	256	900	37.83%	0.4737	2338	16.1
236	64	300	33.62%	0.4429	2332	18.5
108	42	150	26.46%	0.3763	2332	18.8

Table 1: Performance of StreamPETR (Wang et al. 2023) with different query counts. The model uses two query types during inference: queries generated from pre-defined reference points (Ref. Queries) and propagated from Ref. Queries (Pro. Queries). All experiments were run for 24 epochs.

Despite effectiveness, these methods are computationally intensive due to the large number of object queries required. More precisely, the number of pre-defined queries is typically set to 300 for 2D object detection tasks (Meng et al. 2021; Chen et al. 2023; Gao et al. 2022), and this number increases to 900 for 3D detection (Wang et al. 2022; Li et al. 2023; Liu et al. 2022; Wang, Jiang, and Li 2023; Wang et al. 2023; Jiang et al. 2024), which significantly exceeds the actual number of objects in both cases. As depicted in DETR3D (Wang et al. 2022), the performance of the model is positively correlated with the number of queries. We further validated this conclusion through experiments with different query settings in StreamPETR (Wang et al. 2023). As shown in Table 1, reducing the number of queries consistently degrades performance.

Since fewer queries lead to fewer predictions during the inference stage, reducing queries will make the model require additional computation to cover the solution space and may even result in failure to find optimal parameters. Therefore, using fewer queries generally results in poorer performance.

For instance, as illustrated in Figure 1 (a), if a model were to use only a single query to predict two object types – such as pedestrians and barriers – that query would need to cap-

*These authors contributed equally.
†Corresponding author.
Copyright © 2026, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

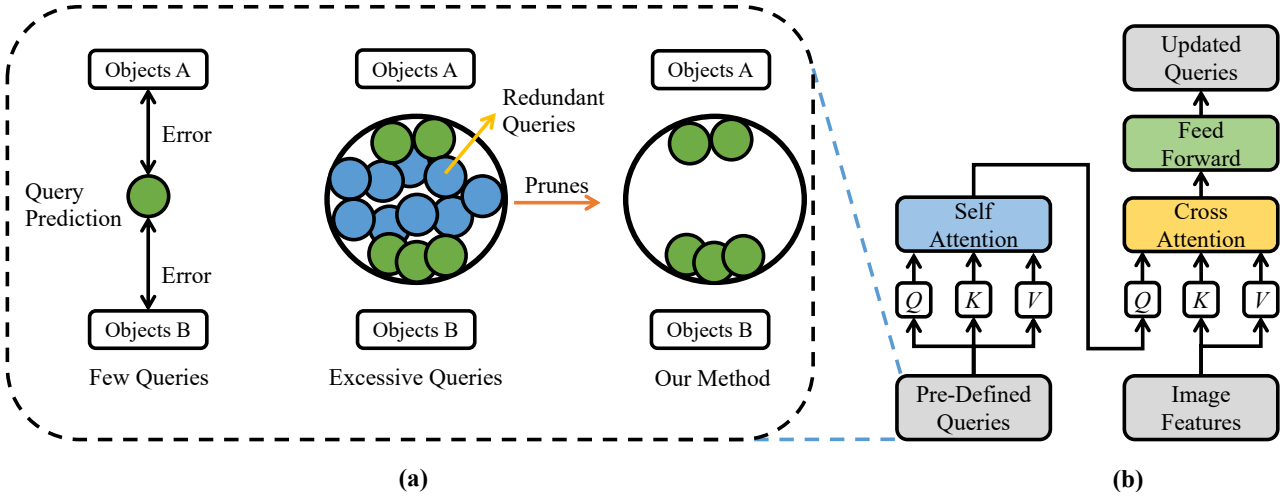

Figure 1: Illustration of query-based detection methods. (a) Using a few queries forces them to cover multiple objects, leading to poor predictions. Introducing excessive queries assigns one per object but introduces redundancy. Our method prunes queries with minimal contribution. (b) The workflow of a single transformer layer. Pre-defined queries are fed into the self-attention module, where they interact with each other. The output of self-attention then serves as the *query* in cross-attention, with image features acting as the *key* and *value*.

ture the features of both objects simultaneously. By handling multiple tasks, the query would need to strike a compromise between the performance of the two object types. If we use more queries, such as two, one can focus on detecting pedestrians while the other identifies barriers, which reduces the interference between the two queries. This means that the more queries there are, the less burden each individual query has to bear. Given the complexity of spatial attributes in 3D detection tasks – such as location, rotation, and velocity for moving objects – it is understandable that the performance of query-based models is sensitive to the number of queries.

1.1 Overloading Queries is Inefficient

As is well known, DETR-based models use the Hungarian algorithm to match predictions with ground-truth labels. When the number of queries far exceeds the number of objects, most predictions are treated as negative instances during bipartite matching. In typical non-specialized scenes, fewer than 100 objects need to be detected (Caesar et al. 2020; Wilson et al. 2021), meaning the negative-to-positive ratio during training can reach 8:1. These negative instances lack ground-truth supervision, so zero vectors are matched with them, progressively pushing their classification scores lower.

When negative instances greatly outnumber positives, query selection becomes increasingly imbalanced. This not only wastes computation but also suppresses the classification scores of queries frequently matched as negatives. During inference, NMS-free selectors choose predictions with the highest classification scores (Carion et al. 2020). For example, using 900 queries to predict 10 classes yields a tensor of size 900×10, i.e., 9,000 instances. The top-k instances are selected as final predictions, making low-scoring queries unlikely to be chosen. As shown in Figure 2, the selection frequency becomes highly uneven, and some queries are never selected. These underutilized queries mainly output background predictions and contribute far less than frequently selected ones.

Given their minimal contribution, can these queries be safely discarded without harming performance?

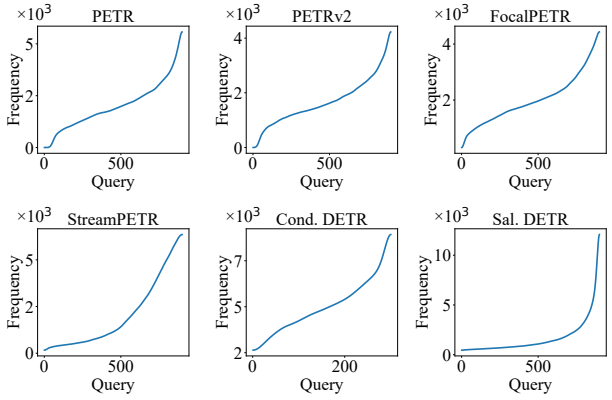

Figure 2: Selection frequency of queries (sorted in ascending order) during inference for different methods. PETR, PETRv2, FocalPETR, and StreamPETR are 3D detectors, while the other two are 2D. As shown, query selection is imbalanced across both 2D and 3D methods. In PETR, PETRv2, and FocalPETR, some queries are never selected as final outputs.

1.2 Pruning Redundant Queries

To address query redundancy, we propose GPQ, a method that gradually prunes redundant queries. Queries with

lower classification scores are deemed less contributive and weaker in representation. Thus, we retain only high-scoring queries to maintain performance while reducing redundancy.

Our method is embarrassingly simple: load a checkpoint and run the model as it should be, then sort the queries by classification scores after each iteration, and remove the bottom-k queries every n iterations. Compared to existing pruning methods that utilize a learnable binary mask (Yu et al. 2022; Kong et al. 2022; Tang et al. 2023; Shi et al. 2023; Chen et al. 2024; Ilhan et al. 2024; Khaki and Plataniotis 2024; Wei et al. 2023; Yu and Xiang 2023), GPQ introduces no additional learnable parameters and, therefore, incurs no extra computational cost. Additionally, our method allows for the creation of multiple model versions with varying numbers of queries using an existing checkpoint that contains a large number of queries. This eliminates the need to retrain the model with fewer queries, which would require additional time to restore performance.

Our contributions can be summarized as follows:

1. To the best of our knowledge, we are the first to address the issue of redundant queries in commonly used 3D object detectors and to conduct a comprehensive analysis of the role of queries in detection transformers. Our findings indicate that the majority of queries in existing query-based methods are redundant and unnecessary.
2. We propose an embarrassingly simple yet effective strategy that gradually prunes redundant queries in detection transformers, enabling us to better utilize existing pre-trained checkpoints to reduce model complexity while maintaining detector performance.
3. We conducted extensive experiments on various query-based detectors to evaluate the effectiveness of our proposed method. The results indicate that, on desktop GPUs, GPQ achieves inference acceleration of up to 1.35× and it achieves at most a 67.87% reduction in FLOPs with a 65.16% decrease in inference time after deployment on edge devices.

2 Related Work

2.1 DETR-Based 3D Object Detectors

Based on transformers (Vaswani et al. 2017), DETR-based methods have been widely used in 3D object detection tasks (Li et al. 2022; Yang et al. 2023; Liu et al. 2022, 2023a; Wang, Jiang, and Li 2023; Jiang et al. 2024; Li et al. 2023; Liu et al. 2023b; Chen et al. 2022; Wang et al. 2024; Xie et al. 2023). (Liu et al. 2022) develops position embedding transformation to generate position-aware 3D features. (Liu et al. 2023a) introduces temporal modeling and generates 3D positional embedding according to the features of input images. (Wang et al. 2023) proposes an object-centric temporal modeling method that combines history information with little memory cost. By utilizing a high-quality 2D object prior, Far3D (Jiang et al. 2024) generates 3D adaptive queries to complement 3D global queries, which extends the detection range to 150 meters.

All these methods use similar pre-defined queries, either as learnable parameters or generated from pre-defined reference points. Despite variations in sampling strategies (Li et al. 2022; Zhu et al. 2020), they all share the *query*, *key*, and *value* components, which are processed through transformer layers. While these methods share similar designs and advantages, they also face common drawbacks: high computational costs and excessive memory usage. This makes it particularly challenging to deploy DETR-based methods on edge devices.

2.2 Transformer Pruning Methods

Distillation and pruning methods have been developed to reduce the resource requirements of large models (Cheng, Zhang, and Shi 2024; Yang et al. 2022; Li et al. 2024; Liang et al. 2023). (Michel, Levy, and Neubig 2019) discovers that a large percentage of heads can be removed at test time. (Fan, Grave, and Joulin 2019) randomly drops transformer layers at training time. (Chen et al. 2021) explores sparsity in vision transformers, which proposes an unstructured and a structured method to prune the weights. While magnitude pruning methods rely on the absolute values of parameters to determine which units to prune, (Sanh, Wolf, and Rush 2020) uses first-order information rather than zero-order information as the pruning criterion. (Lagunas et al. 2021) extends (Sanh, Wolf, and Rush 2020) to local blocks of any size. (Yu et al. 2022) reduces both width and depth of the transformer simultaneously. (Kwon et al. 2022) proposes a post-training pruning method that does not need retraining. (Yu and Xiang 2023) proposes an explainable method by assigning each unit an explainability-aware mask. (Liu et al. 2024b) prunes tokens in the ViT(Dosovitskiy 2020) model, where pruned tokens in the feature maps are preserved for later use.

Recently, retraining-free pruning methods have attracted significant attention. (Bolya et al. 2023) divides image tokens into two groups and merges top-r most similar pairs. (Xu et al. 2025) uses the attention matrix and classification scores to prune keys in transformer decoders. However, retraining-free pruning methods tend to cause a decline in model performance. Moreover, since pruning is performed at runtime, it may make objects undetectable in certain scenarios, thereby increasing the safety risks of the model.

Most existing pruning methods rely on binary masks to identify parameters for removal (Yu and Xiang 2023; Yu et al. 2022; Sanh, Wolf, and Rush 2020; Lagunas et al. 2021), which introduces additional training overhead. Moreover, they often underperform or fail to generalize to 3D object detection due to several fundamental challenges:

Structural Inconsistencies. In NLP and ViT models, input tokens simultaneously serve as queries, keys, and values, forming a square attention map of size $N_q \times N_k$. Many pruning methods rely on this structure (Wang, Dedhia, and Jha 2024). However, object detection models use pre-defined queries, leading to $N_q \neq N_k$ and a non-square attention map, which makes these methods unsuitable for 3D object detection.

Massive Data Differences. In ViT models, each image yields fewer than 200 tokens, while 3D object detection must process multi-view images and predict additional indicators, producing at least 4,000-30,000 tokens depending on resolution and backbone. This large token count poses challenges

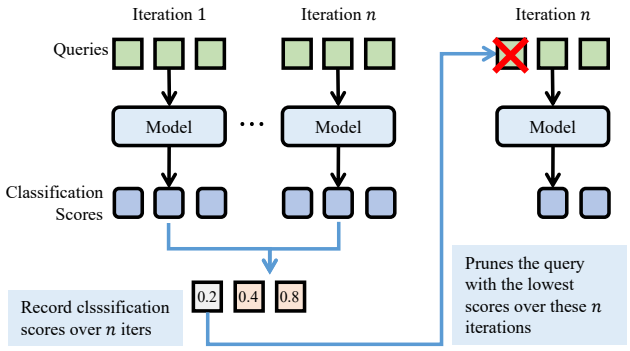

Figure 3: The pruning process. We select the query that generates the lowest classification scores every n iterations. The selected query is then removed from the model and will no longer participate in any operations after being pruned.

for token-pruning methods whose computational cost scales with the number of tokens(Bolya et al. 2023; Wang, Dedhia, and Jha 2024).

These challenges highlight the need for dedicated pruning strategies tailored to the unique characteristics of 3D object detection models.

Algorithm 1: Gradually Pruning Queries (GPQ)

Input: Total iterations T, the number of initial (final) queries N_q (N_q'), a 3D detection model with queries Q, and pruning interval n.

Load the model from an existing checkpoint with queries Q
Initialize current iteration $t = 1$, and the number of current queries $N = N_q$
while $t \leq T$ **do**
  Update Q through Transformer-Layers
  Get classification scores $\mathcal{C}$ through the Classification-Branch
  if $N > N_q'$ **then**
    Record classification scores of each query
  if $t \mod n = 0$ *and* $N > N_q'$ **then**
    Select the query that generates the lowest classification score relying on records
    Remove the selected query from Q
    Update current query number $N \leftarrow N - 1$
  $t \leftarrow t + 1$

Output: Final pruned queries $\mathcal{Q} \leftarrow Q$

3 Method

3.1 Revisiting Query-based Methods

Each transformer layer used in query-based detection methods typically consists of a self-attention layer, a cross-attention layer, and a feed-forward network, as illustrated in Figure 1 (b). The attention (Vaswani et al. 2017) operation contains three inputs: *query* $Q \in \mathbb{R}^{N_q \times E}$, *key* $K \in \mathbb{R}^{N_k \times E}$ and *value* $V \in \mathbb{R}^{N_k \times D_v}$. An attention weight matrix will be calculated using *query* and *key*, which will be used to sample *value*. Learnable parameters in the attention operation are its projection matrices $W^Q \in \mathbb{R}^{E \times E}$, $W^K \in \mathbb{R}^{E \times E}$, $W^V \in \mathbb{R}^{D_v \times D_v}$ and $W^O \in \mathbb{R}^{D_v \times D_v}$:

$$\text{Attn} = \text{Softmax}\left(\frac{(QW^Q)(KW^K)^{\mathrm{T}}}{\sqrt{E}}\right)(VW^V)W^O \quad (1)$$

where N_q is the number of queries, N_k is the number of keys and values, E is the dimension for *query* and *key*, and D_v is the dimension for *value*.

The self-attention module uses the pre-defined queries as *query*, *key*, and *value*, and its output is then fed into the cross-attention module as the *query* to interact with image features. To stack multiple transformer layers, the output of layer l serves as the input for layer $l + 1$, allowing the pre-defined queries to be updated. Let $F_I \in \mathbb{R}^{N_k \times E}$ represent the image features. This process can be described as follows:

$$\begin{aligned} Q_l &\leftarrow \text{Self-Attention}_l(Q_{l-1}) \\ Q_l &\leftarrow \text{Cross-Attention}_l(Q_l, F_I, F_I) \\ Q_l &\leftarrow \text{FFN}_l(Q_l) \end{aligned} \quad (2)$$

where $l \in \{1, 2, \cdots, N\}$ is the index of current transformer layer, N is the number of layers and FFN is the feedforward network, which is a multi-layer perception with a large hidden dimension $h \gg D_v$. Q_0 represents the initial pre-defined queries, which will be updated at each layer through interactions with themselves in self-attention and with image features in cross-attention.

3.2 The GPQ Algorithm

We treat each query as the basic pruning unit and **use classification scores as the pruning criterion**. During the iteration process, we gradually prune redundant queries. The pruning operation is triggered every n iterations. Each time, we select the query that generates the lowest classification scores and remove it from current queries. The dropped query is immediately removed from the model. During both training and inference, these queries will no longer participate in any operations. Note that it is the queries, not the predictions, that are dropped. The pruning procedure is illustrated in Figure 3, and the pseudo-code is provided in Algorithm 1.

3.3 Why is Pruning Queries Effective?

Our method is effective because queries are largely independent of one another—removing a specific query has minimal impact on the rest. This independence can be understood through the rule of matrix multiplication: multiplying a matrix $A \in \mathbb{R}^{N_A \times M}$ by a matrix $B \in \mathbb{R}^{M \times N_B}$ is equivalent to multiplying each row $A_i \in \mathbb{R}^M$ of A by B individually, and then concatenating the results:

$$AB \equiv \underset{i=1,\cdots,N_A}{\text{Concat}}\ (A_i B) \quad (3)$$

where N_A, N_B and M are positive integers. If we delete the i-th row from A, the results of the multiplication involving the remaining rows with B will remain unchanged.

Model	Backbone	Image Size	Queries	FPS↑	mAP ↑	NDS ↑	mATE↓	mASE↓	mAOE↓	mAVE↓	mAAE↓
DETR3D (CORL 2021)	ResNet50	1408x512	900 / -	8.2	24.63%	0.3054	0.9534	0.2867	0.7005	0.9959	0.2417
			300 / -	9.2	23.43%	0.2953	0.9851	0.2865	0.7189	0.9667	0.2613
			150 / -	9.2	20.55%	0.2694	1.0045	0.2925	0.7426	1.1308	0.2981
			900 / 300	9.3	**24.78%**	**0.3234**	**0.9404**	**0.2789**	**0.6139**	**0.9397**	0.2326
			900 / 150	9.4	22.63%	0.3015	0.9713	0.2813	0.6444	0.9919	**0.2276**
PETR (ECCV 2022)	ResNet50	1408x512	900 / -	6.9	31.74%	0.3668	0.8395	0.2797	0.6153	0.9522	0.2322
			300 / -	8.9	31.19%	0.3536	0.8449	0.2872	0.6156	1.0673	0.2762
			150 / -	9.2	28.37%	0.3158	0.8664	0.2899	0.7340	1.1074	0.3706
			900 / 300	8.9	**32.85%**	**0.3884**	**0.8003**	**0.2791**	**0.5507**	**0.9108**	**0.2179**
			900 / 150	9.3	30.52%	0.3671	0.8237	0.2792	0.5804	0.9441	0.2282
	VovNet	1600x640	900 / -	3.1	**40.45%**	**0.4517**	0.7282	0.2706	0.4482	**0.8404**	0.2179
			300 / -	3.8	38.98%	0.4279	0.7636	0.2732	0.4820	0.9198	0.2315
			150/-	3.9	38.80%	0.4436	0.7375	**0.2691**	0.4396	0.8504	**0.2104**
			900 / 300	3.7	40.04%	0.4507	**0.7278**	0.2723	**0.4383**	0.8451	0.2110
			900 / 150	4.0	39.03%	0.4445	0.7324	0.2692	0.4460	0.8450	0.2134
PETRv2 (ICCV 2023)	VovNet	800x320	900 / -	5.5	**40.64%**	**0.4949**	**0.7374**	0.2693	0.4636	0.4162	0.1967
			300 / -	6.5	39.19%	0.4893	0.7595	**0.2678**	**0.4416**	0.4360	0.1916
			150 / -	6.8	38.00%	0.4709	0.7710	0.2760	0.4773	0.4652	0.2013
			900 / 300	6.7	40.26%	0.4944	0.7383	0.2701	0.4542	**0.4146**	**0.1916**
			900 / 150	6.9	39.16%	0.4919	0.7385	.02702	0.4271	0.4135	0.1898
FocalPETR (TIV 2023)	ResNet50	704x256	900 / -	16.4	32.44%	0.3752	0.7458	**0.2778**	0.6489	0.9458	0.2522
			300 / -	19.3	31.59%	0.3524	0.7594	0.2838	0.7154	1.0432	0.2973
			150 / -	21.2	27.78%	0.3071	0.8276	0.2826	0.7863	1.2156	0.4178
			900 / 300	19.6	**33.17%**	**0.3925**	**0.7446**	0.2800	**0.6265**	**0.8619**	**0.2203**
			900 / 150	21.2	31.81%	0.3834	0.7563	0.2829	0.6119	0.8792	0.2259
StreamPETR (ICCV 2023)	ResNet50	704x256	900 / -	16.1	37.83%	0.4734	0.6961	0.2822	0.6846	0.2856	0.2084
			300 / -	18.5	33.62%	0.4429	0.7305	0.2837	0.6800	0.3333	0.2251
			150 / -	18.8	26.46%	0.3763	0.8195	0.2921	0.8135	0.3998	0.2353
			900 / 300	18.7	**39.42%**	**0.4941**	**0.6766**	**0.2711**	**0.5799**	**0.2708**	0.2136
			900 / 150	19.3	34.94%	0.4633	0.6989	0.2749	0.6226	0.3124	**0.2050**
RayDN (ECCV 2024)	ResNet50	704x256	900 / -	15.3	**45.93%**	0.5455	**0.5880**	0.2666	0.5144	0.2674	0.2042
			300 / -	16.8	43.31%	0.5255	0.6020	0.2728	0.5526	0.2737	0.2093
			900 / 300	17.0	45.71%	**0.5507**	0.5923	**0.2636**	**0.4652**	**0.2590**	**0.1989**

Table 2: Pruning results across models. The column of "Queries" is of format "# Initial queries/ # Final queries". Lines where "# Final queries" remain blank are baselines used to compare. All baselines and checkpoints are trained for 24 epochs, with the pruning process completed within the first 6 epochs when loading a checkpoint. FPS is measured on a single RTX3090.

In each MLP and cross-attention module, the query matrix Q appears only once. This means that, according to Equ. 3, if we remove Q_i from Q, the results of the other queries in the MLP and cross-attention modules will remain unaffected.

The only influence occurs in the self-attention modules. In the self-attention mechanism, the *query* Q also serves as both the *key* K and the *value* V. When a query Q_i is removed, the other queries are affected because the right side of the matrix multiplication has also changed. According to Equ. 1, self-attention mechanism can be formated as:

$$\text{SA} = \text{Softmax}\left(\frac{(QW^Q)(QW^K)^{\mathrm{T}}}{\sqrt{E}}\right)(QW^V)W^O \quad (4)$$

In multi-layer transformers, the queries interact with image features in the cross-attention module, so queries in the deeper layers of the transformer also contain feature information related to the input image during self-attention. At this stage, self-attention can partially replicate the function of cross-attention by sampling image features. However, this sampling is indirect and has less impact compared to cross-attention. Therefore, we can eliminate these queries.

3.4 Why not Train a Model with Fewer Queries?

We propose a query pruning method to reduce the computational load of DETR-based detectors. A natural question is: why not directly train models with fewer queries? As discussed in Section 1, training with more queries improves the model's ability to handle diverse objects. With GPQ, a checkpoint trained with many queries can be pruned to different query counts, providing flexibility across scenarios without retraining for each configuration. More details are provided in the extended version.

4 Experiments

4.1 Experimental Setup

Dataset and Detectors We conduct our experiments on the nuScenes dataset (Caesar et al. 2020), which consists of over 23,000 samples. Each sample includes images from six surrounding cameras, covering the full 360° field of view. The 3D detection task involves 10 object classes (car, truck, bus, trailer, construction vehicle, pedestrian, motorcycle, bicycle, barrier, and traffic cone), including both static and dynamic objects.

To validate the effectiveness and efficiency of our proposed method, we perform experiments on five advanced detectors: DETR (Wang et al. 2022), PETR (Liu et al. 2022), PETRv2 (Liu et al. 2023a), FocalPETR (Wang, Jiang, and Li 2023), and StreamPETR (Wang et al. 2023). Notably, PETRv2 employs VovNet (Lee and Park 2020) as its image backbone, while the others utilize ResNet50 (He et al. 2016) as their image backbone.

Evaluation Metrics In the field of 3D object detection, the primary performance evaluation metrics are mean Average Precision (mAP). NuScenes also provides the nuScenes Detection Score (NDS), which is derived from mAP, along with several other error metrics: mean Average Translation Error (mATE), mean Average Scale Error (mASE), mean Average Orientation Error (mAOE), mean Average Velocity Error (mAVE), and mean Average Attribute Error (mAAE). Mathematically, $\text{NDS} = \frac{1}{10}\left[5\text{mAP} + \sum_{\text{mTP}} (1 - \min(1, \text{mTP}))\right]$, where $\text{mTP} \in \{\text{mATE}, \text{mASE}, \text{mAOE}, \text{mAVE}, \text{mAAE}\}$.

Additionally, to assess the improvement in model runtime achieved by our method, we calculate the FLOPs (Floating Point Operations) for each module and record the model's runtime before and after pruning on resource-constrained edge devices. Typically, we use GFLOPs, where $1\ \text{GFLOPs} = 1 \times 10^9\ \text{FLOPs}$.

Model	Backbone	Pruned	# Queries	GFLOPs	Time (ms)
PETR	ResNet18	×	900	219.14	1829.44
		✓	300	164.61(-24.89%)	1231.46(-32.69%)
		✓	150	152.06(-30.61%)	1103.96(-39.66%)
	w/o	×	900	99.39	1140.97
		✓	300	44.86(-54.87%)	563.85(-50.58%)
		✓	150	32.30(-67.50%)	439.34(-61.49%)
FocalPETR	ResNet18	×	900	162.36	1319.35
		✓	300	118.07(-27.28%)	846.62(-35.83%)
		✓	150	108.08(-33.44%)	745.44(-43.50%)
	w/o	×	900	78.07	868.28
		✓	300	33.77(-56.75%)	416.12(-52.07%)
		✓	150	23.77(-69.55%)	319.84(-63.16%)
StreamPETR	ResNet18	×	900	172.08	1520.07
		✓	300	123.90(-28.00%)	916.03(-39.74%)
		✓	150	112.51(-34.62%)	791.08(-47.96%)
	w/o	×	900	87.78	1030.38
		✓	300	39.59(-54.90%)	477.81(-53.63%)
		✓	150	28.21(-67.86%)	359.00(-65.16%)

Table 3: Results on Jetson Nano. Models were exported to ONNX format for deployment on Jetson Nano.

4.2 Main Results

Detection Performance: Before vs. After Pruning One of the key objectives of our pruning method is to maintain the performance of the original models. As demonstrated in Table 2, our approach successfully preserves, and in some cases enhances, the performance of various detectors, even when pruning a checkpoint with a large number of queries. GPQ can also accelerate model inference on desktop GPUs. For example, in the case of PETR, pruning from 900 to 150 queries results in an mAP of 30.52%, which is 2.15 points higher than training from scratch with 150 queries (28.37%), and its speed increases from 6.9 fps to 9.3, which is 1.35x faster. Remarkably, pruning from 900 to 300 queries achieves an mAP of 32.85%, outperforming the result of training from scratch with 900 queries (31.74%). Similarly, for both FocalPETR and StreamPETR, pruning from 900 to 300 queries yields optimal performance, surpassing the results obtained from training with 900 queries from scratch. For PETRv2 and RayDN(Liu et al. 2024a), while pruning from 900 to 300 queries results in a slightly lower mAP than training from scratch with 900 queries, it still exceeds the performance of training from scratch with 300 queries.

For 3D objects, their poses and moving states are also important. The NDS of pruned models with a final count of 300 is higher than that of models with 900 queries for PETR, FocalPETR, and StreamPETR. In the case of PETRv2, although the NDS slightly decreases for pruned models, it remains comparable to the performance with 900 queries. Moreover, pruned models consistently outperform those trained from scratch with the same number of queries.

In summary, the results in Table 2 show that GPQ significantly reduces the number of queries while maintaining, or even slightly improving, performance compared to models trained from scratch with a larger number of queries. We also provide visualization results in the extended version.

Inference Speed: Before vs. After Pruning Inference speed is crucial for deploying models on edge devices. To verify whether pruning queries can indeed improve speed, we deploy models with GPQ on the Jetson Nano B01 to measure the model's running time. As shown in Table 3, after pruning from 900 to 300 queries, the FLOPs of StreamPETR are reduced by 28%, and the running time decreases by 39.74%. Pruning further from 900 to 150 queries results in a 47.96% reduction in running time, making the model $1.92\times$ faster.

Since our method does not modify the image backbones, we remove the backbone modules to better illustrate the effects of our approach. Specifically, we run only the transformer decoder using randomly generated dummy inputs. For all the evaluated models, pruning from 900 queries to 300 results in saving more than half of the running time for transformer-related modules. Additionally, compared to the model with 900 queries, StreamPETR with 150 pruned queries achieves a 65.16% reduction in running time. The pruning method also proves effective for FocalPETR and PETR, as shown in Table 3.

Comparison We applied GBC (Xu et al. 2025) and ToMe (Bolya et al. 2023) to StreamPETR and compared their performance with GPQ in Table 4. Surprisingly, ToMe slows down the model instead of accelerating it. This is likely due to the overhead of computing the similarity ma-

Pruning	r	Queries	Speed(fps)	mAP ↑	NDS ↑	mATE ↓	mASE ↓	mAOE ↓	mAVE ↓	mAAE ↓
-	-	900/-	16.1	37.83%	0.4734	0.6961	0.2822	0.6846	0.2856	0.2084
ToMe (ICLR 2023)	2112	900/-	16.0	31.69%	0.4325	0.7546	0.2907	0.6893	0.3170	0.2210
GBC (ICCV 2025)	2000	900/-	**19.3**	37.93%	0.4817	0.6787	0.2758	0.6390	0.2844	**0.2016**
GPQ	-	900/300	18.7	**39.42%**	**0.4941**	**0.6766**	**0.2711**	**0.5799**	**0.2780**	0.2136

Table 4: Comparison with ToMe and GBC. Results indicate that ToMe performs poorly on 3D object detection, while our GPQ effectively maintains the model's performance and accelerates inference. r indicates keys pruned by GBC or ToMe.

Model	mAP ↑	NDS ↑	mATE ↓	mASE ↓	mAOE ↓	mAVE ↓	mAAE ↓
Base	37.83%	0.4734	0.6961	0.2822	0.6846	0.2856	**0.2084**
GPQ-H	34.34%	0.4563	0.7429	0.2813	0.5912	0.3188	0.2195
GPQ-C	38.78%	0.4899	0.6808	0.2814	0.5791	0.2853	0.2130
GPQ-1	35.71%	0.4677	0.7121	0.2828	0.5970	0.2930	0.2233
GPQ	**39.42%**	**0.4941**	**0.6766**	**0.2711**	**0.5799**	**0.2780**	0.2136

Table 5: Ablation experiments use StreamPETR as the baseline. All pruning strategies start from a checkpoint with 900 queries and progressively reduce them to 300.

trix. Unlike image classification tasks, which typically handle only a few hundred tokens, 3D object detection methods often generate a much larger number of tokens (e.g., StreamPETR generates 4224 tokens even with a relatively small image size of 704×256). This increased token size significantly amplifies the computational cost of constructing the similarity matrix, making ToMe inefficient for 3D object detection tasks. Different from ToMe, GBC can achieve a faster speed. However, compared to GPQ, it will lead to a performance drop. Compared to ToMe and GBC, GPQ both achieve better performance and faster speed.

4.3 Ablation Studies

Pruning Criterion GPQ prunes queries with the lowest classification scores. To validate this choice, we conducted two comparative experiments: one where we prune queries with the highest classification scores (**GPQ-H** in Table 5) and another where pruning is guided by the cost produced by the assigner during the binary matching process between predicted and ground truth values (**GPQ-C** in Table 5).

Table 5 shows that pruning queries with the highest classification scores leads to a noticeable performance drop compared to our default strategy. While using the cost generated by the assigner as the pruning criterion results in performance closer to the original model, it still falls short of the performance achieved by GPQ. These results confirm the effectiveness of our proposed method.

Pruning Strategy A key feature of our method is the gradual pruning strategy. To validate its effectiveness, we conducted an experiment where 600 queries were pruned in a single iteration (**GPQ-1** in Table 5). The results show a significant performance drop when all queries are pruned in a single step instead of gradually. This demonstrates that the gradual pruning strategy employed in GPQ is not only reasonable but also the optimal approach for query pruning.

4.4 Results on 2D Object Detection Task

Although our focus is on 3D object detection, 2D object detection remains a fundamental task in computer vision. To assess the broader applicability of GPQ, we further evaluate it on 2D detection tasks.

As noted in the Introduction, queries in 3D detection models are highly redundant, while 2D DETR-based methods typically use about 300 queries (Carion et al. 2020; Meng et al. 2021) to predict 80 classes (e.g., COCO (Lin et al. 2014)), resulting in lower redundancy. This explains why GPQ primarily targets 3D detection. Nevertheless, since DETR-based models also employ pre-defined queries, GPQ remains applicable to 2D detection.

We tested GPQ on ConditionalDETR. As shown in Table 6, pruning half of the queries preserved the original performance, while requiring only 8 epochs—far less than the 50 epochs needed for retraining from scratch. These results demonstrate that GPQ can also be effectively applied to 2D object detection.

Queries	FPS	mAP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
300/-	18.6	0.409	0.618	0.434	0.206	0.442	0.591
150/-	23.3	0.398	0.606	0.421	0.196	0.432	0.582
300/150	23.5	0.406	0.615	0.429	0.197	0.439	0.598

Table 6: Results of GPQ applied to ConditionalDETR. The original ConditionalDETR model uses 300 queries and is trained for 50 epochs. In our approach, we prune half of these queries in just 2 epochs and then fine-tune for an additional 6 epochs.

Additional results can be found in the extended version, including visualization results, evaluations on fully converged models, and integration within the training process.

5 Conclusion

In this paper, we introduced GPQ, a simple yet highly effective method that gradually removes redundant queries in DETR-based 3D detection models. The goal is to reduce computational costs without sacrificing performance. Our results demonstrate that GPQ effectively maintains detection performance across all evaluated models. Additionally, when deployed on the edge device, the pruned models have shown a significant acceleration of inference speed, making them more suitable for real-world applications.

To our knowledge, this is the first study to explore query pruning in query-based models. We hope that our work will inspire further research into pruning DETR-based models.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant No. 62036008, and by the National Key R&D Program of China under Grant No. 2022ZD0117903.

References

Bolya, D.; Fu, C.-Y.; Dai, X.; Zhang, P.; Feichtenhofer, C.; and Hoffman, J. 2023. Token Merging: Your ViT But Faster. In *The Eleventh International Conference on Learning Representations*.

Caesar, H.; Bankiti, V.; Lang, A. H.; Vora, S.; Liong, V. E.; Xu, Q.; Krishnan, A.; Pan, Y.; Baldan, G.; and Beijbom, O. 2020. nuScenes: A Multimodal Dataset for Autonomous Driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Carion, N.; Massa, F.; Synnaeve, G.; Usunier, N.; Kirillov, A.; and Zagoruyko, S. 2020. End-to-end object detection with transformers. In *European Conference on Computer Vision*, 213–229. Springer.

Chen, Q.; Chen, X.; Wang, J.; Zhang, S.; Yao, K.; Feng, H.; Han, J.; Ding, E.; Zeng, G.; and Wang, J. 2023. Group detr: Fast detr training with group-wise one-to-many assignment. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 6633–6642.

Chen, T.; Cheng, Y.; Gan, Z.; Yuan, L.; Zhang, L.; and Wang, Z. 2021. Chasing sparsity in vision transformers: An end-to-end exploration. *Advances in Neural Information Processing Systems*, 34: 19974–19988.

Chen, Z.; Li, Z.; Zhang, S.; Fang, L.; Jiang, Q.; and Zhao, F. 2022. Deformable feature aggregation for dynamic multi-modal 3D object detection. In *European Conference on Computer Vision*, 628–644. Springer.

Chen, Z.; Li, Z.; Zhang, S.; Fang, L.; Jiang, Q.; and Zhao, F. 2024. Deformable feature aggregation for dynamic multi-modal 3D object detection. In *European Conference on Computer Vision*.

Cheng, H.; Zhang, M.; and Shi, J. Q. 2024. A Survey on Deep Neural Network Pruning: Taxonomy, Comparison, Analysis, and Recommendations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12): 10558–10578.

Dosovitskiy, A. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Fan, A.; Grave, E.; and Joulin, A. 2019. Reducing transformer depth on demand with structured dropout. *arXiv preprint arXiv:1909.11556*.

Fan, L.; Yang, Y.; Wang, F.; Wang, N.; and Zhang, Z. 2023. Super Sparse 3D Object Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(10): 12490–12505.

Fang, R.; Gao, P.; Zhou, A.; Cai, Y.; Liu, S.; Dai, J.; and Li, H. 2024. FeatAug-DETR: Enriching One-to-Many Matching for DETRs With Feature Augmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(9): 6402–6415.

Gao, Z.; Wang, L.; Han, B.; and Guo, S. 2022. Adamixer: A fast-converging query-based object detector. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5364–5373.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Ilhan, F.; Su, G.; Tekin, S. F.; Huang, T.; Hu, S.; and Liu, L. 2024. Resource-Efficient Transformer Pruning for Finetuning of Large Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 16206–16215.

Jiang, X.; Li, S.; Liu, Y.; Wang, S.; Jia, F.; Wang, T.; Han, L.; and Zhang, X. 2024. Far3d: Expanding the horizon for surround-view 3d object detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2561–2569.

Khaki, S.; and Plataniotis, K. N. 2024. The Need for Speed: Pruning Transformers with One Recipe. In *The Twelfth International Conference on Learning Representations*.

Kong, Z.; Dong, P.; Ma, X.; Meng, X.; Niu, W.; Sun, M.; Shen, X.; Yuan, G.; Ren, B.; Tang, H.; et al. 2022. Spvit: Enabling faster vision transformers via latency-aware soft token pruning. In *European Conference on Computer Vision*, 620–640. Springer.

Kwon, W.; Kim, S.; Mahoney, M. W.; Hassoun, J.; Keutzer, K.; and Gholami, A. 2022. A fast post-training pruning framework for transformers. *Advances in Neural Information Processing Systems*, 35: 24101–24116.

Lagunas, F.; Charlaix, E.; Sanh, V.; and Rush, A. M. 2021. Block Pruning For Faster Transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 10619–10629.

Lee, Y.; and Park, J. 2020. Centermask: Real-time anchor-free instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 13906–13915.

Li, X.; Yin, J.; Li, W.; Xu, C.; Yang, R.; and Shen, J. 2024. Di-v2x: Learning domain-invariant representation for vehicle-infrastructure collaborative 3d object detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 3208–3215.

Li, Y.; Ge, Z.; Yu, G.; Yang, J.; Wang, Z.; Shi, Y.; Sun, J.; and Li, Z. 2023. Bevdepth: Acquisition of reliable depth for multi-view 3d object detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 1477–1485.

Li, Z.; Wang, W.; Li, H.; Xie, E.; Sima, C.; Lu, T.; Qiao, Y.; and Dai, J. 2022. Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. In *European Conference on Computer Vision*, 1–18. Springer.

Liang, C.; Zuo, S.; Zhang, Q.; He, P.; Chen, W.; and Zhao, T. 2023. Less is more: Task-aware layer-wise distillation for language model compression. In *International Conference on Machine Learning*, 20852–20867. PMLR.

Liao, B.; Chen, S.; Wang, X.; Cheng, T.; Zhang, Q.; Liu, W.; and Huang, C. 2023a. MapTR: Structured Modeling and Learning for Online Vectorized HD Map Construction. In *International Conference on Learning Representations*.

Liao, B.; Chen, S.; Zhang, Y.; Jiang, B.; Zhang, Q.; Liu, W.; Huang, C.; and Wang, X. 2023b. MapTRv2: An End-to-End Framework for Online Vectorized HD Map Construction. *arXiv preprint arXiv:2308.05736*.

Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; and Zitnick, C. L. 2014. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, 740–755. Springer.

Liu, F.; Huang, T.; Zhang, Q.; Yao, H.; Zhang, C.; Wan, F.; Ye, Q.; and Zhou, Y. 2024a. Ray Denoising: Depth-aware Hard Negative Sampling for Multi-view 3D Object Detection. In *European Conference on Computer Vision*. Springer.

Liu, Y.; Gehrig, M.; Messikommer, N.; Cannici, M.; and Scaramuzza, D. 2024b. Revisiting token pruning for object detection

and instance segmentation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2658–2668.

Liu, Y.; Wang, T.; Zhang, X.; and Sun, J. 2022. Petr: Position embedding transformation for multi-view 3d object detection. In *European Conference on Computer Vision*, 531–548. Springer.

Liu, Y.; Yan, J.; Jia, F.; Li, S.; Gao, A.; Wang, T.; and Zhang, X. 2023a. PETRv2: A Unified Framework for 3D Perception from Multi-Camera Images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 3262–3272.

Liu, Z.; Tang, H.; Amini, A.; Yang, X.; Mao, H.; Rus, D. L.; and Han, S. 2023b. Bevfusion: Multi-task multi-sensor fusion with unified bird's-eye view representation. In *2023 IEEE international conference on robotics and automation (ICRA)*, 2774–2781. IEEE.

Ma, X.; Ouyang, W.; Simonelli, A.; and Ricci, E. 2024. 3D Object Detection From Images for Autonomous Driving: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(5): 3537–3556.

Meng, D.; Chen, X.; Fan, Z.; Zeng, G.; Li, H.; Yuan, Y.; Sun, L.; and Wang, J. 2021. Conditional DETR for Fast Training Convergence. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 3631–3640.

Michel, P.; Levy, O.; and Neubig, G. 2019. Are sixteen heads really better than one? *Advances in neural information processing systems*, 32.

Sanh, V.; Wolf, T.; and Rush, A. 2020. Movement pruning: Adaptive sparsity by fine-tuning. *Advances in neural information processing systems*, 33: 20378–20389.

Shi, D.; Tao, C.; Jin, Y.; Yang, Z.; Yuan, C.; and Wang, J. 2023. Upop: Unified and progressive pruning for compressing vision-language transformers. In *International Conference on Machine Learning*, 31292–31311. PMLR.

Tang, Q.; Zhang, B.; Liu, J.; Liu, F.; and Liu, Y. 2023. Dynamic token pruning in plain vision transformers for semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 777–786.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Wang, H.; Dedhia, B.; and Jha, N. K. 2024. Zero-TPrune: Zero-shot token pruning through leveraging of the attention graph in pre-trained transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 16070–16079.

Wang, J.; Meng, Q.; Liu, G.; Yan, L.; Wang, K.; Cheng, M.-M.; and Hou, Q. 2025. Towards stable 3d object detection. In *European Conference on Computer Vision*, 197–213. Springer.

Wang, S.; Jiang, X.; and Li, Y. 2023. Focal-petr: Embracing foreground for efficient multi-camera 3d object detection. *IEEE Transactions on Intelligent Vehicles*.

Wang, S.; Liu, Y.; Wang, T.; Li, Y.; and Zhang, X. 2023. Exploring object-centric temporal modeling for efficient multi-view 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 3621–3631.

Wang, Y.; Guizilini, V. C.; Zhang, T.; Wang, Y.; Zhao, H.; and Solomon, J. 2022. Detr3d: 3d object detection from multi-view images via 3d-to-2d queries. In *Conference on Robot Learning*, 180–191. PMLR.

Wang, Z.; Li, Y.-L.; Chen, X.; Zhao, H.; and Wang, S. 2024. Uni3detr: Unified 3d detection transformer. *Advances in Neural Information Processing Systems*, 36.

Wei, S.; Ye, T.; Zhang, S.; Tang, Y.; and Liang, J. 2023. Joint token pruning and squeezing towards more aggressive compression of vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2092–2101.

Wilson, B.; Qi, W.; Agarwal, T.; Lambert, J.; Singh, J.; Khandelwal, S.; Pan, B.; Kumar, R.; Hartnett, A.; Pontes, J. K.; Ramanan, D.; Carr, P.; and Hays, J. 2021. Argoverse 2: Next Generation Datasets for Self-driving Perception and Forecasting. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS Datasets and Benchmarks 2021)*.

Xie, Y.; Jiang, H.; Gkioxari, G.; and Straub, J. 2023. Pixel-aligned recurrent queries for multi-view 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 18370–18380.

Xu, L.; Bai, X.; Jia, X.; Fang, J.; and Pang, S. 2025. Accelerate 3D Object Detection Models via Zero-Shot Attention Key Pruning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Yang, C.; Chen, Y.; Tian, H.; Tao, C.; Zhu, X.; Zhang, Z.; Huang, G.; Li, H.; Qiao, Y.; Lu, L.; Zhou, J.; and Dai, J. 2023. BEVFormer v2: Adapting Modern Image Backbones to Bird's-Eye-View Recognition via Perspective Supervision. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 17830–17839.

Yang, Z.; Li, Z.; Jiang, X.; Gong, Y.; Yuan, Z.; Zhao, D.; and Yuan, C. 2022. Focal and global knowledge distillation for detectors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4643–4652.

Yu, F.; Huang, K.; Wang, M.; Cheng, Y.; Chu, W.; and Cui, L. 2022. Width & depth pruning for vision transformers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 3143–3151.

Yu, L.; and Xiang, W. 2023. X-pruner: explainable pruning for vision transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 24355–24363.

Zhu, X.; Su, W.; Lu, L.; Li, B.; Wang, X.; and Dai, J. 2020. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*.