Appendix

Anonymous authors

A Motivation Example for the Study

Our study is driven by the need to develop a query tool for point cloud data to address specific tasks. We use a real-world example to further illustrate the importance and necessity of this query tool, reinforcing the motivation behind our research.

Querying specific frames in point cloud data is crucial for facilitating target tasks. For example, to enhance a model's ability to detect pedestrians, a direct and effective approach is to train or fine-tune the model using frames containing pedestrians, rather than relying on the entire dataset. This targeted approach not only improves performance but also reduces computation costs. We conducted a simple experiment using the KITTI dataset. As shown in Figure 1, we compared pedestrian detection performance under three conditions: (1)Training on the full dataset (3712 frames). (2)Training on a randomly sampled subset of 1,000 frames. (3)Training on a carefully selected subset of frames containing pedestrians (about 955 frames).

The results demonstrated that using the carefully selected subset improved pedestrian detection performance, achieving similar or better results compared to the full dataset while reducing training costs by approximately two-thirds.

This experiment highlights the necessity of querying point cloud data, proving its potential to enhance task-specific performance and optimize resource utilization.

	Car			Pedestrian			Cyclist		
	Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard
Full (3712 frames)	69.42	61.44	58.33	33.88	31.96	31.25	63.47	48.28	46.11
Random (1000 frames)	70.72	60.31	55.56	35.38	33.11	31.91	60.04	44.26	42.57
Pedestrain (955 frames)	70.38	58.38	54.63	**37.88**	**36.11**	**34.53**	59.46	43.81	42.49

Table 1: Detection results of training with different set

B Data Model Design and Explanation for Query Type Selection

B.1 Data Model

As illustrated below, our data model is designed with NoSQL format, which brought more flexibility in aggregating data from different sources, such as different vehicles, making it more suitable for handling the diverse and complex nature of autonomous driving data. Object counting, being one of the most essential pieces of information required for querying [Kang *et al.*, 2022; Kang *et al.*, ; Cao *et al.*, 2022; Bang *et al.*, 2023], shapes the design of our data model, as shown below. We store key information from each frame, including object type, count, and the approximate positions of objects in a bird's eye view, ensuring that the data model is optimized for efficient querying.

```
{ "frame_id": "frame123",
  "timestamp": "2024-04-07
     15:43:02.8924030000",
  "vehicle_id": "vehicle_00000000",
  "objects": [
         { "type": "car",
            "count": "10",
         "position":[{"x": 1.5, "y": 2.1},
     ...]},
      ...]
}
```

B.2 Query Type Selection

The query we defined cover a wide range of common analytical needs and practical use cases. To be more specific, SELECT query is fundamental for retrieving specific frames that meet certain criteria, which is for tasks like object tracking and scene analysis. COUNT query provides a quick way to quantify how many frames meet a specific condition. This is useful for statistical analysis, monitoring, and quality control, such as ensuring a balanced dataset or evaluating the frequency of certain events or objects. AGGREGATION Query is essential for summarizing data across multiple frames, which helps in understanding overall trends, performing long-term analysis, and feeding summarized data into higher-level systems for further processing or decision-making. JOIN Query allows for combining information across multiple conditions or object types within the same dataset. This is particularly useful when complex, multi-faceted analyses are required, such as understanding scenarios involving multiple types of objects and their interactions.

C Methodology

C.1 Thresholding for Heatmap

During training, we set the default threshold to 0.5. Since the heatmap values are normalized within the range of 0 to 1, selecting a mid-range value for the threshold ensures effective filtering. Intuitively, the model is expected to assign higher values (above 0.5) to regions corresponding to object grids, while non-object regions will have lower values (below 0.5).

Otsu's Method for Dynamic Thresholding

In Section **??**, we use Otsu's Method for Dynamic Thresholding. Specifically, Otsu's method maximizes the between-class variance, given by the formula: $\alpha_B^2(k) = \omega_0(k) \times \omega_1(k) \times [m_0(k) - m_1(k)]^2$. Here, $w_0(k)$ and $w_1(k)$ are the proportion of points below and above the threshold, respectively, while $m_0(k)$ and $m_1(k)$ represent the mean values of the points below and above k, respectively. By maximizing this function, we find the optimal threshold k for each partition.

C.2 Overlapped Partition

Algorithm 1: Inference with Overlap Partition

Input: feature map F, expanding ratio δ, radius γ
Output: centers

1 $R = getPartitions(F)$;
// Expand regions
2 **foreach** $r \in R$ **do**
3 $expand\_w = r.width \times \delta$;
4 $expand\_h = r.height \times \delta$;
5 $r.x_{start} = max(0, r.x_{start} - expand\_w)$;
6 $r.y_{start} = max(0, r.y_{start} - expand\_h)$;
7 $r.x_{end} = min(F.width, r.x_{end} + expand\_w)$;
8 $r.y_{end} = min(F.height, r.y_{end} + expand\_h)$;
9 $H = getHeatmaps(R)$;
10 $map = zeros((F.width, F.height))$// init map of 0
11 **foreach** $h \in H$ **do**
12 C = getCentersFromHeatmap(h) ;
// region coord -> map coord
13 C = transCoordinate(C) ;
14 **foreach** $c \in centers$ **do**
15 map[c.x, c.y] = 1
// Combine duplicate centers
16 $centers = getCenters(map)$;
17 $new\_centers = \emptyset$;
18 **while** *centers* **do**
19 $current\_center = centers.pop(0)$;
20 $new\_centers.add(current_c)$;
21 $centers = combineCenters(centers, r)$;
22 return $new\_centers$

The training process and loss calculation follow the same procedure as CounterNet with Partition introduced in Section **??**. However, the inference process involves additional steps to account for overlapping partitions, as shown in Figure **??** and Algorithm 1: (1) Generate Expanded Partitioned Heatmaps. Given a feature map, we first divide it into partitions (Line 1). Based on the expansion ratio δ, we calculate the required width and height for the expansion (Lines 3-4) and determine the region coordinates with the expanded width and height (Lines 5-8). (2)Extract Centers from Partitions and Merge. We generate the heatmap for each partitioned region (Line 9). Afterward, the centers extracted from each heatmap are merged into a single map (Lines 10-15). (3) Remove Duplicate Center Points. Using a predefined radius γ, we merge centers that fall within this radius (Lines 16-21).

D Additional Experiments

D.1 Evaluation Metrics

(1) SELECT. We experimented with two types of SELECT queries: SELECT-COUNT and SELECT-BINARY. SELECT-COUNT query retrieved frames containing a specific number of objects. We evaluated it by calculating the percentage of frames where the object count was correctly determined, allowing for a 10% and 5% error rate. SELECT-BINARY query checked if a frame contained any objects or not. For each object category, we calculated the percentage of frames that were correctly identified.

(2) COUNT. The COUNT query was used to count the number of frames that satisfied a specific condition, such as counting the frames containing 5 cars. To evaluate this, we randomly selected 500 groups of consecutive frames with lengths ranging from 100 to 500. For each object category, we generated 1000 random queries requesting different object numbers (which do not exceed the maximum number of objects present in any frame) and evaluated the percentage of counts being returned correctly. We allowed a 10% error rate in the results.

(3) AGGREGATION. The Aggregation query provided statistical information, such as the total number of objects in a set of frames. We evaluated this using the SUM() operation. For random consecutive frame groups of lengths between 100 and 500, we calculated the sum of the queried objects and compared the evaluated sum to the ground truth across 1000 random queries. The performance was measured by the average difference between the ground truth and evaluated sums.

(4) JOIN. To evaluate the JOIN query, we applied SELECT-COUNT to combinations of different object categories. The performance was measured by the percentage of frames that were correctly counted for multiple objects, allowing for a 10% error rate.

Evaluate Object Counting. The evaluation count result can be reflected from the result of SELECT-COUNT, which evaluate by the percentage of frames that we can correctly count the object number.

D.2 Ablation Studies

Dynamic Thresholding. As explained in Section **??**, we proposed using dynamic thresholding to filter out noise for peak detection. We compared the results with and without dynamic thresholding for the model CN-p4o. In our evaluation, we selected three fixed thresholds: 0.5, 0.52, and 0.54, for comparison. As shown in Figure 1, using a fixed threshold resulted in significant deviations across object categories. For instance, the motorcycle and bicycle categories experienced extremely low performance with thresholds of 0.5 and 0.52,

# of objects	1-5	6-10	11-15	16-20	21-30	31-40	41-50	¿51
car	1953	1794	1070	578	223	69	19	8
pedestrian	2863	724	231	197	113	30	6	12
barrier	463	329	280	190	113	33	16	4
traffic_cone	1408	417	104	48	71	10	1	–
truck	3225	248	63	6	–	–	–	–
bus	1504	12	–	–	–	–	–	–
trailer	818	81	37	6	–	–	–	–
construction_vehicle	1134	–	–	–	–	–	–	–
motorcycle	1192	40	–	–	–	–	–	–

Table 2: Statistic of nuScenes dataset

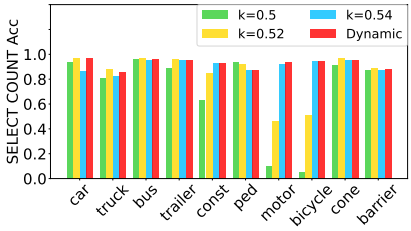

Figure 1: Result of Dynamic Thresholding

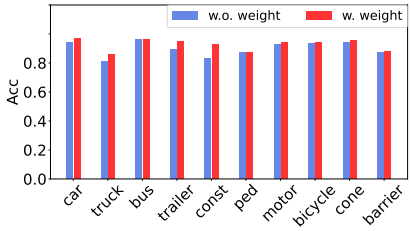

Figure 2: Result of Weighted Loss with Partitioning

while for the car and truck categories, a threshold of 0.54 performed poorly. In contrast, dynamic thresholding provided a more balanced performance across all categories, achieving consistently good results.

Weighted loss. In Section **??**, we introduced the weighted loss during the model training with partitions. We introduce the weighted loss intent to improve the prediction with the frame with more objects appear. By refer to the data statistics of nuScenes dataset in Appendix D.3, we can see that the weighted loss contributed more to the low freqnecy object, such as truck, trailer and construction vechicle. While for the high frequency appearance object, such as car, pedestrian and barrier, the improvement with partition is limited.

D.3 Data Distribution for nuScenes Dataset

Table 2 demonstrates the data distribution of the nuScenes dataset. In this table, we present the statistics showing the number of frames that contain a specific number of objects (as indicated in the first row).

D.4 Object Counting Accuracy and Recall

Table 3 summarizes the performance of the object counting task by evaluating both accuracy and recall. Accuracy reflects the overall performance across all frames, while recall emphasizes performance on frames containing objects, considering the prevalence of frames without objects.

The following observations can be made: (1) For the "car" category, the recall rate closely aligns with the accuracy, showing minimal deviation and indicating consistent performance across metrics. (2) When the error rate tolerance is reduced from 10% to 5%, the proposed method exhibits smaller performance fluctuations compared to baseline methods, demonstrating its robustness under stricter error constraints. (3) Partitioning slightly degrades overall performance; however, it improves performance for objects that frequently appear in large quantities within a single frame, such as cars. (4) Incorporating overlaps improves the overall recall rate as it enhances coverage of overlapping regions, enabling more effective object detection.

D.5 More Experiments for Model Selection

Our full experiment for evaluating model selection includes: (1) **Evaluation of Different Model Combinations.** We conduct experiments using different combinations of models. The results for 9 models and 6 models are presented in Table 4 and Table 5, respectively. The corresponding counting results are illustrated in Figure 3 and Figure 4.
(2) **Weight Adjustment for Specific Categories.** When generating the center of each model, we assign higher weights to the categories of "cars" and "pedestrians". The results of this adjustment are shown in Table 6, with the corresponding counting performance in Figure 5.

With the experiments, we have the following observations: (1) **Compare results of using 6 models and 9 models.** Table 5 and Figure 4 show the results of using 6 models for model selection. Compared to the results with 9 models, we removed the dominant models from the 9-model set (i.e., p4, p4-o0.1, and p4-o0.2). According to Table 5, after removing the p4 series models, the new dominant model without adjustment becomes p9. However, with the adjustment, the dominant model shifts to p9-o0.1, while p2-o0.1 remains the second dominant model. From the results in Figure 4, it is evident that applying adjustments dramatically improves accuracy. When comparing the adjusted results to the best-performing model (p4-o0.2), even the weaker-performing models achieve comparable results through model selection with adjustments.
(2) **Impact of Weight Adjustment on Object Categories.** We also evaluated the effect of adjusting weights for different objects in model selection. Table 6 displays the results when the weights for the "car" and "pedestrian" categories are doubled, compared to the even-weight approach shown in Table **??**. From Table 6, the dominant model remains p4-o0.2; however, the second dominant model shifts from p4 to p9-o0.2. As shown in Figure 5, this adjustment significantly

		car	truck	bus	trailer	const	ped	moto	bicyc	cone	barrier
Acc (10% error rate)	VN	0.801	0.778	0.917	0.915	0.845	0.711	0.728	0.632	0.706	0.761
	TF	0.615	0.621	0.959	0.925	0.886	0.640	0.867	0.828	0.722	0.802
	CN	0.807	0.802	0.957	0.953	0.929	0.870	0.933	0.940	0.933	0.874
	CN(p4)	0.919	0.799	0.944	0.950	0.910	0.869	0.886	0.898	0.926	0.883
	CN(p4o)	0.967	0.858	0.960	0.951	0.931	0.873	0.940	0.942	0.954	0.883
Acc (5% error rate)	VN	0.536	0.594	0.917	0.815	0.849	0.469	0.728	0.632	0.528	0.640
	TF	0.506	0.610	0.959	0.909	0.886	0.553	0.866	0.907	0.717	0.755
	CN	0.601	0.659	0.957	0.912	0.929	0.741	0.933	0.940	0.875	0.811
	CN(p4)	0.774	0.609	0.944	0.880	0.909	0.735	0.886	0.898	0.874	0.823
	CN(p4o)	0.861	0.705	0.960	0.871	0.810	0.748	0.940	0.942	0.867	0.824
Recall (10% error rate)	VN	0.791	0.708	0.883	0.745	0.792	0.617	0.692	0.840	0.772	0.599
	TF	0.839	0.740	0.927	0.702	0.760	0.540	0.849	0.811	0.793	0.579
	CN	0.877	0.777	0.898	0.816	0.781	0.849	**0.873**	**0.865**	0.858	0.665
	CN(p4)	0.915	0.754	0.845	0.780	0.795	0.827	0.828	0.778	0.806	0.626
	CN(p4o)	0.965	0.811	0.895	0.852	0.799	0.850	0.845*	0.821*	0.909	0.694
Recall (5% error rate)	VN	0.519	0.513	0.883	0.611	0.772	0.672	0.692	0.610	0.635	0.401
	TF	0.493	0.511	0.905	0.571	0.760	0.600	0.849	0.811	0.574	0.398
	CN	0.709	0.577	0.898	0.644	0.781	0.688	**0.873**	**0.865**	0.735	0.523
	CN(p4)	0.765	0.545	0.845	0.613	0.795	0.660	0.828	0.778	0.678	0.444
	CN(p4o)	0.856	0.631	0.894	0.707	0.778	0.675*	0.845*	0.821*	0.790	0.584

Table 3: Accuracy and Recall for Object Counting

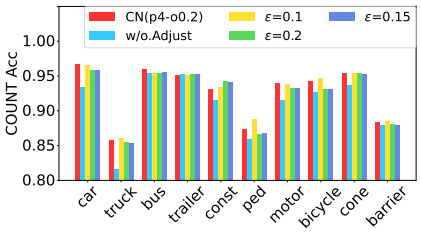

Figure 3: Result of model selection with 9 models

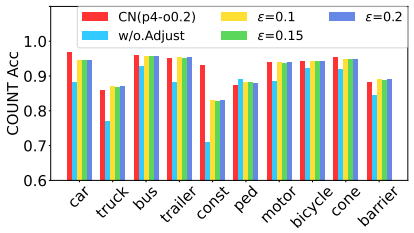

Figure 4: Result of model selection with 6 models

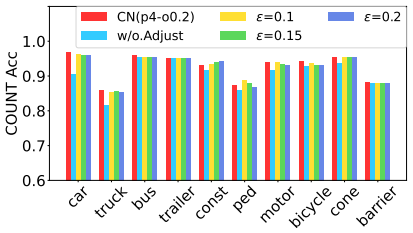

Figure 5: Result of model selection with more weight on car and pedestrian

no adjustment		$\epsilon = 0.1$		$\epsilon = 0.15$		$\epsilon = 0.2$	
Model	dist	rate	dist	rate	dist	rate	dist
p2	1116	0.835	339	0.667	168	0.486	90
p2-o0.1	21	0.780	279	0.572	274	0.371	343
p2-o0.2	48	0.811	43	0.624	35	0.433	32
p4	9	0.543	556	0.379	743	0.087	752
p4-o0.1	14	0.650	9	0.379	7	0.178	8
p4-o0.2	4739	0.617	4736	0.337	4734	0.145	4735
p9	20	0.642	0	0.426	3	0.219	3
p9-o0.1	34	0.693	48	0.369	53	0.170	54
p9-o0.2	16	0.684	7	0.439	35	0.231	0

Table 4: Data distribution of model selection with adjustment with 9 models ("dist" is short for distribution)

no adjustment		$\epsilon = 0.1$		$\epsilon = 0.15$		$\epsilon = 0.2$	
Model	dist	rate	dist	rate	dist	rate	dist
p2	0	0.718	0	0.475	0	0.266	0
p2-o0.1	1171	0.631	641	0.355	561	0.159	713
p2-o0.2	60	0.633	54	0.358	53	0.161	56
p9	6	0.616	0	0.336	0	0.144	1
p9-o0.1	4752	0.405	5323	0.131	5404	0.026	5237
p9-o0.2	30	0.497	1	0.207	1	0.061	12

Table 5: Data distribution of model selection with adjustment with 6 models

improves performance for the "car" and "pedestrian" categories, while maintaining stable performance for other object categories.

D.6 Results on KITTI dataset

We evaluate our proposed method on the KITTI dataset. We use CenterPoint [Yin *et al.*, 2021] (CP) as the baseline detection model for comparison.

no adjustment		$\epsilon = 0.1$		$\epsilon = 0.15$		$\epsilon = 0.2$	
Model	dist	rate	dist	rate	dist	rate	dist
p2	19	0.833	14	0.663	12	0.481	9
p2-o0.1	1133	0.809	656	0.621	461	0.429	414
p2-o0.2	20	0.837	18	0.670	17	0.490	17
p4	23	0.660	22	0.393	24	0.190	59
p4-o0.1	22	0.634	17	0.358	13	0.161	14
p4-o0.2	4728	0.578	4734	0.292	4735	0.112	4734
p9	26	0.628	19	0.352	18	0.156	19
p9-o0.1	13	0.593	502	0.309	702	0.124	717
p9-o0.2	33	0.691	35	0.435	35	0.228	34

Table 6: Data distribution of model selection with adjustment with car and pedestrian preferred

Evaluation of Query Results

As shown in Table 8 and Table 9, the performance improvement of our proposed CounterNet (CN) on the KITTI dataset is less pronounced compared to the improvements observed on the nuScenes dataset when compared to the detection model CenterPoint (CP). For the SELECT-COUNT operation, while the performance improvements for the "car" and "cyclist" categories are not substantial, a noticeable improvement is observed for the "pedestrian" category.

Regarding partitioning and overlap, the results for our proposed solution are consistent with those observed on the nuScenes dataset. However, considering that the scene scale of nuScenes is approximately four times larger than that of KITTI, partitioning has a more negative impact on performance in the KITTI dataset. Nevertheless, the use of overlap partially mitigates this negative effect.

Query Information			ACC					Recall					Precision				
Query	Predict	Selectivity	VN	TF	CN	CN-p4	CN-p4o	VN	TF	CN	CN-p4	CN-p4o	VN	TF	CN	CN-p4	CN-p4o
Q1	SELECT(car>5)	67.9%	0.812	0.768	0.614	0.778	**0.850**	0.815	0.769	0.996	0.972	0.929	0.995	0.997	0.615	0.796	0.908
Q2	SELECT(barrier>3)	76.2%	0.639	0.748	0.859	**0.866**	0.764	0.954	0.953	0.905	0.893	0.934	0.641	0.752	0.943	0.967	0.807
Q3	SELECT(bus>0)	33.2%	0.488	0.507	**0.553**	0.445	0.557	0.504	0.628	0.898	0.762	0.759	0.940	0.949	0.591	0.515	0.632
Q5	JOIN(car>5, ped>0)	31.5%	0.635	0.604	0.520	0.624	**0.702**	0.638	0.605	0.967	0.857	0.841	0.962	0.966	0.529	0.697	0.809
Q6	JOIN(truck>0, barrier>5)	16.8%	0.404	0.448	0.332	0.465	**0.529**	0.411	0.452	0.761	0.882	0.678	0.964	0.977	0.371	0.496	0.706

Table 7: Case study of different query scenario on nuScenes

	SELECT-BINARY				SELECT-COUNT(5% error rate)				AGGREGATION			
	CP	CN	CN(p4)	CN(p4o)	CP	CN	CN(p4)	CN(p4o)	CP	CN	CN(p4)	CN(p4o)
car	**0.952**	0.9156	0.929	0.934	0.692	0.705	0.681	**0.731**	**0.135**	0.280	2.25	0.172
ped	0.602	0.675	0.639	**0.792**	0.776	0.778	0.781	**0.828**	0.704	**0.167**	3.07	0.447
cyclist	0.760	0.797	0.773	**0.812**	0.901	0.925	0.922	**0.931**	0.152	**0.121**	2.183	0.211

Table 8: Query result on KITTI dataset

	CP	CN	CN(p4)	CN(p4o)
(car, pedestrian)	0.812	0.856	0.854	0.902
(car, cyclist)	0.885	0.887	0.880	0.903
(pedestrian, cyclist)	0.833	0.886	0.868	0.926
(car, pedestrian, cyclist)	0.776	0.821	0.811	0.869

Table 9: Result of JOIN query on KITTI

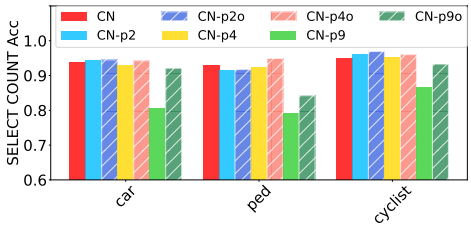

Figure 6: Result of different number of partitions on KITTI

no adjustment		ϵ = 0.05		ϵ = 0.1		ϵ = 0.15	
Model	dist	rate	dist	rate	dist	rate	dist
p2	937	0.956	0	0.838	0	0.672	0
p2-o0.2	597	0.555	94	0.095	1052	0.005	1648
p4	512	0.775	0	0.362	0	0.101	61
p4-o0.2	165	0.307	3675	0.008	2717	0.00002	2060
p9	1525	0.965	0	0.868	0	0.728	0
p9-o0.2	33	0.890	0	0.628	0	0.351	0

Table 10: Data distribution of model selection with adjustment on KITTI

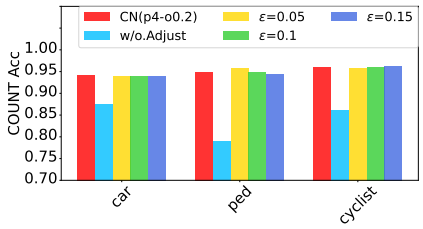

Figure 7: Result of model selection on KITTI

Evaluation of Partition and Overlap

Figure 6 illustrates the results of partitioning with 2, 4, and 9 partitions, as well as the corresponding results when overlap is applied (with a default overlap ratio of 0.2). The following observations can be made: (1) When using partitioning alone, partition 2 (CN-p2) outperforms partition 4 (CN-p4) and partition 9 (CN-p9) in most cases, except for the "pedestrian" category. This is likely due to the smaller size of pedestrian objects, which benefit more from finer partitions. (2) Given the smaller scene scale of the KITTI dataset, using partition 9 significantly decreases performance across all categories. (3) By introducing overlaps, larger partitions benefit more from overlap. For instance, the performance improvement with overlap is greater for partition 4 (CN-p4o) and partition 9 (CN-p9o) compared to partition 2 (CN-p2o).

Evaluation of Model Selection

Table 10 and Figure 7 present the results of model selection with six models on the KITTI dataset. Overall, the findings are consistent with those observed on the nuScenes dataset. As shown in Table 10, after applying the adjustment, the frames are primarily concentrated on four partitions with a 0.2 overlap rate (p4-o0.2) and two partitions with a 0.2 overlap rate (p2-o0.2). Furthermore, as illustrated in Figure 7, similar to the nuScenes results, the model selection yields balanced performance across all categories.

E Case studies

We evaluate the performance of the actual query across different scenarios and levels of selectivity on the nuScenes dataset. The experimental results assess accuracy, precision, and recall. The key observations are as follows: (1) Accuracy. Our proposed method consistently outperforms the baseline methods in terms of accuracy. However, when precision and recall are considered, the performance varies depending on the specific query. (2) Selectivity. Overall, performance is better in high-selectivity scenarios, whereas low-selectivity cases introduce more errors across all methods. (3) Query-Specific Analysis. For Query 1 (Q1), CounterNet exhibits a tendency toward higher recall compared to precision, whereas the baselines demonstrate the opposite trend. This indicates that CounterNet achieves higher accuracy within the queried frames but may fail to select some relevant frames. In contrast, the baseline methods tend to select a larger number of frames, capturing more true-positive frames. However, their lower precision indicates reduced accuracy among the selected frames. A similar trend is observed across all other queries.

References

[Bang *et al.*, 2023] Jaeho Bang, Gaurav Tarlok Kakkar, Pramod Chunduri, Subrata Mitra, and Joy Arulraj. Seiden: Revisiting query processing in video database systems. *VLDB*, 16(9):2289–2301, 2023.

[Cao *et al.*, 2022] Jiashen Cao, Karan Sarkar, Ramyad Hadidi, Joy Arulraj, and Hyesoon Kim. Figo: Fine-grained query optimization in video analytics. In *SIGMOD*, pages 559–572, 2022.

[Kang *et al.*,] Daniel Kang, Peter Bailis, and Matei Zaharia. Blazeit: Optimizing declarative aggregation and limit queries for neural network-based video analytics. *VLDB*, 13(4).

[Kang *et al.*, 2022] Daniel Kang, John Guibas, Peter D Bailis, Tatsunori Hashimoto, and Matei Zaharia. Tasti: Semantic indexes for machine learning-based queries over unstructured data. In *SIGMOD*, pages 1934–1947, 2022.

[Yin *et al.*, 2021] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *CVPR*, pages 11784–11793, 2021.