What the nuScenes-QA answers encode: a label audit, and direct answers against executed programs

Nassim Arifette
MVA — Nuages de Points et Modélisation 3D (NPM3D)

November 27, 2026

Abstract

This report makes two contributions on nuScenes-QA [3], a question-answering benchmark over driving scenes. The first is an audit of its released answers. They do not distinguish left from right: with a the unsigned angle between the ego heading and the direction to an object, "front left" holds for $30° < a \leq 90°$ on either side, "back right" for $90° < a \leq 150°$ on either side, and "front right" and "back left" hold for no object. All 12,392 counting or existence questions of the official train and validation sets that name one of the last two relations are answered 0 or *no*. "Things" include every annotation, the classes no question names among them, and the ego vehicle; boxes with no LiDAR return are counted. On 1,460 questions from 146 validation drives, a deterministic executor agrees with the released answers on 85.1% under the published reading and on 97.5% under these three conventions. The left/right rule was found on other drives; the "things" rule was found on these and confirmed on the train set. The second contribution is a comparison of two uses of one frozen language model on ground-truth object lists: answering directly, or writing a JSON program that a deterministic executor runs. Our earlier comparison stated the published convention to both modes and scored both against labels that follow another one. Executing the same programs under the convention of the labels moves six models from 0.685–0.845 to 0.765–0.980 on 200 questions from three drives; the plateau near 0.845 was the mismatch. On the 146 validation drives, with both modes told the conventions of the labels, Qwen2.5-7B reaches 0.961 with programs against 0.595 directly, a difference of +0.366 with a 95% drive-level bootstrap interval of $[+0.337, +0.395]$; Qwen2.5-1.5B reaches 0.747 against 0.582 and Qwen2.5-32B (4-bit) 0.978 against 0.679. Asking the direct mode to write out its working did not close the gap for the 1.5B and 7B models; Qwen3-8B with its thinking mode on is right on every development question it finishes within 4,096 tokens, and finishes 63 of 100. The rules are inferred from the answers, not read from the generation code, and perception is an oracle throughout. The result concerns these question families; it does not show that language models cannot reason about space.

1 Introduction

nuScenes-QA [3] pairs nuScenes keyframes [2] with templated questions about what is around the ego vehicle: how many, whether any, in which of six directions, with which status. Systems such as LiDAR-LLM [1] are evaluated on it, and a number on its directional questions is usually read as a measure of spatial reasoning.

This work started as a course project that reimplemented a reduced LiDAR-LLM and could not train it on the hardware available (Section 2.1). The project then fixed perception and compared two uses of one frozen language model: *direct*, where the model reads an object list and answers, and *program*, where it writes a small JSON program and a deterministic executor computes the answer from the same list. A first version of this report presented that comparison under the relations as the benchmark paper defines them, six signed 60° sectors. In that version

the program mode stopped near 0.845 for every model but the smallest, and we attributed the executor's remaining disagreements with the reference answers to objects near sector boundaries.

That attribution was wrong. The disagreements came from a convention of the labels that the published definition does not state. This revision is organised around what followed.

Contribution 1: an audit of the labels (Section 3). Three conventions of the released answers differ from what the paper's text and the question wording suggest: left and right are not distinguished; "things" are every annotation plus the ego vehicle; objects with no LiDAR return are counted. Each is stated as a rule and tested on data that played no part in finding it, including the official train set.

Contribution 2: direct against program once the conventions are right (Section 5). The archived comparison scored both modes against labels that follow a convention neither was told. We re-execute the archived programs under the convention of the labels, rerun the direct mode with a prompt that states it, add an ablation where the direct mode writes out its working, treat constrained decoding as a separate condition, and repeat the comparison on 146 validation drives with intervals that resample drives. The archived numbers stay in Section 4, labelled as results under the published reading.

Limits, stated once here and again where they apply. Perception is an oracle: object lists are ground-truth annotations. The first comparison has three evaluation drives. Prompts and the layout were developed on nuScenes v1.0-mini. The left/right rule was found by looking at evaluation failures and then confirmed on held-out data; the "things" rule was found on the validation drives and confirmed on the train set. The rules describe the answers; the generation code was not read. The 32B model is quantised to 4 bits.

2 Setting

2.1 Related work and origin of the project

CLEVR [4] represents a question as a functional program over a small catalogue of operations (filter, unique, relate, count, exist, query, compare) executed on a scene graph, and neural-symbolic VQA [5] showed that recovering and executing such a program is far more accurate on CLEVR than predicting the answer end to end. The comparison in this report reproduces that finding on real driving annotations with frozen models and no training; the idea is not ours. Closer in domain, ZSVG3D [6] generates programs for zero-shot 3D grounding and KLDrive [7] combines scene facts with constrained LLM reasoning. Their perception and protocols differ from ours, so they give no comparable numbers. Real-3DQA [8] shows that on SQA3D a model fine-tuned with no 3D input matches or beats existing 3D-LLMs, and that filtering blind-answerable questions drops LEO's exact match from 49.4% to 14.3%; the scene-dependence control of Section 5.7 follows that argument.

The first version of the project implemented LiDAR-LLM [1] in reduced form on 323 training samples, without a GPU. It showed that gradients flow and nothing more, and its evaluation had no question-only control and split questions rather than scenes. The present protocol was designed against those two weaknesses.

2.2 Scenes, programs and the executor

A scene is a list of objects, each with a class (the ten nuScenes detection classes), a centre, a box size, a yaw and, where the annotation carries one, a *status*: moving, parked, stopped, standing, sitting or lying down, with rider, without rider. Coordinates are in the ego frame, x forward,

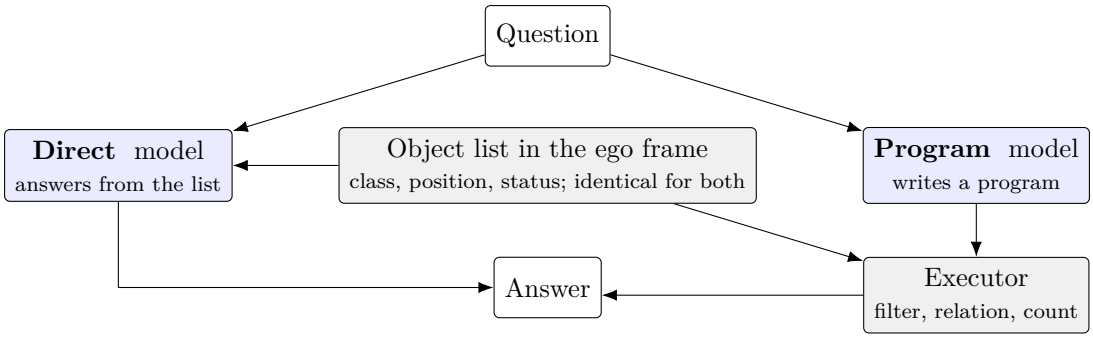

Figure 1: The two modes. Grey is fixed and shared; blue is the same frozen checkpoint used two ways.

y left. An unknown status is treated as neither the attribute nor its negation: a traffic cone is neither "standing" nor "not standing".

A program is an operation (COUNT, EXISTS, COMPARE) over a selector: a category or category group, an optional status, an optional region (one of the six relations, near, far, or a radius), and an optional anchor that makes the region relative to another object. Two parsers emit this language. A deterministic rule grammar produces reference programs and declines what it cannot represent; the frozen language model is the system under test.

Execution is a filter followed by an arithmetic operation. It has no learned parameters. A definite reference whose resolution is ambiguous ("the truck" when the scene has three) is refused, not resolved by taking the first match. Refusals, invalid programs and unreadable answers count as wrong everywhere in this report. Objects closer than 1 m to the ego origin are excluded from directional queries. Relations relative to an object ("to the front of the bus") keep the ego vehicle's heading and move the origin to that object.

The executor takes the spatial convention as a named *layout*. Two matter here: `nuscenes_qa`, the six signed 60° sectors of Equation 2 of the benchmark paper, which we call the *published* reading, and `nuscenes_qa_as_labelled`, the rule of Section 3.1, which we call *as labelled*. A program names a relation and not a layout, so the same generated program can be executed under either.

2.3 Protocol

First comparison. One set of questions was fixed before either mode ran, in the manifest `first_comparison.json`: 100 development and 200 evaluation questions from nuScenes-QA on v1.0-mini, split by drive (seven development drives, three evaluation drives, no overlap). Questions are selected by what they ask, from the surface structure of the text, and not by what a parser manages: 1,626 of the 5,916 mini questions are in scope. The scope is what the rule grammar can express, because the grammar supplies the reference programs against which the executor is checked before any model runs. There are four families: category only ("are there any cars"), category with a status ("how many parked cars"), category in a region relative to the ego vehicle, and category anchored on another object.

Validation drives. A second set, `val_drives.json`, was built in November 2026 by the same rules from the official nuScenes-QA validation questions on the v1.0-trainval annotations: 1,460 questions, ten from each of 146 drives, none of which is in v1.0-mini. It has no development part; development remains the development split of the first comparison.

Matched conditions. Both modes use the same checkpoint through one runtime, the same precision and greedy decoding. The direct mode receives the object rows (id, category, x, y, status) in an order that does not depend on the question, with no pre-filtering and no precomputed

counts. Both prompts state the ego frame and the treatment of unknown status; the direct prompt also states the spatial convention in prose, exactly as the executor computes it. Prompt `direct-1` states the published signed sectors and `direct-2` the as-labelled rule. Prompts longer than the context budget are refused, not trimmed, and every generation records whether it stopped on its own; in the archived runs, at a 256-token budget, no generation was truncated.

Oracle perception. Object lists are ground-truth annotations, not detections. This fixes perception so that the comparison is about what each mode does with a scene. Nothing here speaks to detection quality.

3 An audit of the nuScenes-QA answers

Each convention below was found while explaining disagreements between the executor and the reference answers, then tested on data that played no part in finding it. They are descriptions of the released labels.

3.1 The spatial relations do not distinguish left from right

Let $a \in [0°, 180°]$ be the unsigned angle between the ego heading and the direction to the object. In the released answers, an object is to the *front* when $a \leq 30°$; to the *front left* when $30° < a \leq 90°$, on either side of the vehicle; to the *back right* when $90° < a \leq 150°$, on either side; to the *back* when $a > 150°$. No object is ever to the *front right* or to the *back left*. This is what the arccosine of a normalised dot product, the form of Equation 2 of the benchmark paper, gives when the sign of the angle is never recovered.

Evidence without geometry. Take every counting or existence question of the official release that names exactly one relation, and count how often the answer is 0 or *no* (Table 1). For four relations the rate is between 20% and 24%. For "front right" and "back left" it is 100%: 12,392 questions across 34,117 keyframes, and not one has an object there. The same holds on v1.0-mini (37/37 and 41/41).

		official train		official val	
Relation in the question	True in the labels when	questions	0/no	questions	0/no
front	$a \leq 30°$	19,345	20.8%	4,427	20.2%
front left	$30° < a \leq 90°$, either side	19,243	22.8%	4,156	22.0%
front right	never	5,070	**100%**	1,134	**100%**
back left	never	5,137	**100%**	1,051	**100%**
back right	$90° < a \leq 150°$, either side	19,373	23.4%	4,103	23.9%
back	$a > 150°$	19,778	20.3%	4,284	21.2%

Table 1: The as-labelled rule, and the share of counting or existence questions naming exactly one relation whose answer is 0 or *no*. No scene geometry is used.

Evidence with geometry, on data not used to find the rule. The rule was read off evaluation failures of the first comparison, so it is checked elsewhere (Table 2). On the development split, the hand-verified reference programs agree with the answers on 91 of 100 questions under signed sectors and on 99 under the rule. The one disagreement left is a question whose definite reference ("the not standing pedestrian") matches 22 objects; the dataset answers it anyway and the executor refuses. On the official validation set, executing every program the rule grammar produces gives 82.8% agreement under signed sectors and 91.9% as labelled. The gain is where the rule predicts: "front right" and "back left" go to 100%, "front left" from 53.4% to 91.4%,

"back right" from 57.2% to 90.3%, and the axial relations, which the rule leaves unchanged, do not move. An earlier pass over the directional evaluation questions found that no rotation or mirror of the frame explains the disagreements: the best reached 71 of 97, the unsigned rule 94.

Check	n	signed sectors	as labelled
Mini development split, hand-verified reference programs	100	91	99
Mini evaluation split, same procedure (rule found here)	200	168	195
Official validation set, rule grammar, all it answers	25,973	82.8%	91.9%
of which front left	2,638	53.4%	91.4%
front right	1,092	73.7%	100%
back left	1,018	75.0%	100%
back right	2,535	57.2%	90.3%
front, back	5,804	87.7%	87.7%

Table 2: Agreement between executed reference programs and the released answers under each layout. The validation-set rows use rule-grammar programs that were not read by hand, so 91.9% is a floor on agreement with the labels, not a measurement of label noise.

What the first version of this report said instead. On the development split the executor under signed sectors agreed on 50/50 non-directional questions, 17/18 axial ones and 24/32 diagonal ones, with "front left" alone at 5/10. We described this as a residual at the boundaries of the diagonal sectors and categorised eight of the nine development disagreements that way. That categorisation is superseded: those eight are the left/right convention, and the ninth is the non-unique reference above. An archived diagnostic on 68 ego-anchored evaluation questions had compared signed candidates only (uniform 60° sectors 76.5%, mirrored 72.1%, camera frusta 67.6%, octants 61.8%); it could not find a rule outside the family it searched.

3.2 "Things" are every annotation, and the ego vehicle is one

For "How many things are there?" the answer equals the number of annotations of the keyframe, of every category, plus one. This holds for all 2,080 such questions of the official train set and all 648 of the validation set. The alternatives fail: every annotation without the extra one matches no question in either set, the ten named classes plus one match 1,610 and 495, and annotations with a LiDAR return plus one match 973 and 304. A "thing" therefore includes classes that no question names (pushable objects, debris, bicycle racks, animals), includes boxes with no LiDAR return, and includes the ego vehicle. The ego vehicle also counts in relations between objects ("how many things are to the back of the bus"), though never to the front or back *of me*. This was found on the validation drives of Section 2.3 and confirmed on the train set, which had not been looked at before.

The object-list option `things="all"` implements it: unnamed classes are kept as `other`, the ego vehicle is appended at the origin, and both match only "thing" or "object". Every archived run used the ten named classes; on v1.0-mini the difference is one question in 300. Table 3 gives the executor's agreement with rule-grammar reference programs on the 1,460 validation-drive questions under the four combinations.

Object list	signed sectors	as labelled
Ten named classes	85.1%	92.8%
Every annotation and the ego vehicle	89.6%	**97.5%**

Table 3: Executor agreement with the released answers on the 1,460 questions of the 146 validation drives, reference programs from the rule grammar, no model involved.

3.3 Objects with no LiDAR return are counted

nuScenes records how many LiDAR points fall inside each annotated box. On v1.0-mini, 4,278 of 18,538 annotations (23%) have none, and the answers count them. Holding the programs fixed and changing only the object list gives Table 4. The first block is the table of the first version of this report. It was produced by an earlier version of the rule parser and cannot be regenerated exactly from the current code; the other rows are the current code, under each layout. The conclusion is the same in every row: keeping only objects the sensor returned roughly halves accuracy on counting questions. This is a mismatch between the labels and a LiDAR-only input, not a measured detector error. Every experiment below uses the unfiltered list.

Counting questions both conditions answer	Layout	≥ 1 return only	every annotation
v1.0-mini, 125 (earlier parser, archived)	signed	0.336	0.824
v1.0-mini, 249	signed	0.373	0.799
v1.0-mini, 249	as labelled	0.442	0.940
Official val, 10,087	signed	0.370	0.725
Official val, 10,087	as labelled	0.420	0.827

Table 4: Accuracy with the same programs and geometry, changing only whether annotations with zero LiDAR points are kept.

3.4 What this means for a user of the benchmark, and limits

A system can score well on the directional questions with no notion of left and right, and a system that has one is penalised: a correct count of the cars to the front right is marked wrong whenever there is one. A LiDAR-only system is asked to count objects its sensor did not return and objects of classes a ten-class detector does not have. Results reported on nuScenes-QA remain comparable with each other, since every system faces the same labels. What changes is what a number means: it measures agreement with these conventions, not left/right spatial reasoning. A method that encodes the published geometry, as our executor did, is capped well below 100% for a reason unrelated to its quality.

The limits are these. The rules are inferred from answers; reading the generation code could confirm or refine them. The validation-set check with geometry uses unverified rule-grammar programs. Some questions still disagree under all three conventions: 4 of 200 on the mini evaluation split and 36 of 1,460 on the validation drives. On the validation drives these are 19 directional questions, mostly counts off by one or two, 8 questions whose reference object is not unique or not found, 7 status questions and 2 plain counts. They were not investigated further.

4 Direct against program under the published reading (archived)

This section keeps the results of the first version of the report. Both modes were told the published signed sectors (prompt `direct-1` and the `nuscenes_qa` layout), and both were scored against labels that follow the rule of Section 3.1. The numbers are correct as measurements under that reading. Their interpretation changes in Section 5.

For Qwen2.5-1.5B (Table 5) the program mode is ahead on both splits. Paired by question on the evaluation split it is alone correct on 67 questions and alone wrong on 31 (sign test $p = 3.5 \times 10^{-4}$); on development, 38 against 15. The sign test assumes independent questions, and two hundred questions over three drives are not independent. Per drive the margin is +0.212, +0.188 and +0.138. A bootstrap over three drives can draw only ten distinct resamples, so no interval was reported; Section 5.5 replaces this argument. The direct model under-counts: where the reference is 7, 9 or 10 it answers 1, 3 and 2, and its mean under-count on evaluation is 1.84 against an over-count of 0.31. Its answer format was not the obstacle: all three hundred direct

	development (n=100)		evaluation (n=200)	
	direct	program	direct	program
Accuracy (all)	0.510	**0.740**	0.505	**0.685**
Answers	1.000	0.860	1.000	0.910
Accuracy when answering	0.510	0.860	0.505	0.753
Counting: numeric coverage	1.000	0.946	1.000	0.988
Counting: mean absolute error	2.43	**0.54**	2.16	**0.68**

Table 5: Qwen2.5-1.5B-Instruct, published reading. The evaluation split shares no drive with development.

outputs parsed. On evaluation the program mode reads 178,402 input tokens against 217,272, because it never sees the objects, and writes 9,731 against 400.

			program, executed under		
Model	run	direct-1	published	as labelled	invalid
Qwen2.5-1.5B	local, fp16	0.505	0.685	0.765	8
Qwen2.5-7B	local, fp16	0.505	0.825	0.950	1
Qwen2.5-32B	local, int4	0.580	0.845	0.980	0
`qwen-3-32b`	hosted	0.560	0.840	0.975	0
`qwen3-next-80b`	hosted	0.610	0.845	0.980	0
`qwen3.8-27b`	hosted, reasons	0.840	0.845	0.980	0

Table 6: The 200 evaluation questions of the first comparison, six models. The *direct-1* and *published* columns are the archived results. The *as labelled* column executes the same archived programs under the convention of the labels (Section 5.1). "Invalid" counts outputs that are not programs. The 32B checkpoint is quantised to 4 bits; without a matched full-precision run the effect of quantisation is unknown.

Table 6 adds two larger checkpoints of the same family and three models reached through a hosted gateway. For the hosted rows, `temperature=0` is requested of the endpoint, token counts are the provider's, the names are provider aliases and not immutable checkpoints, and Qwen3-Next-80B-A3B activates 3B of its 80B parameters [9]. Under the published reading we drew three conclusions. Two of them do not survive.

The direct mode gains little with size. It goes from 0.505 to 0.580 across a twenty-one-fold increase in parameters, and the 7B model, at the same 0.505 as the 1.5B, answers differently on 89 of the 200 questions. This observation stands, and Section 5.2 repeats it with the corrected prompt.

The program mode plateaus near 0.845. The five other models land between 0.825 and 0.845, and we wrote that the remaining errors were no longer about writing a well-formed program and pointed to the executor and the data. The plateau is real under this reading: 28 of the 200 questions are wrong in program mode for all six models, so no model could pass 0.860. It is not a property of the task or of program writing. It is the convention mismatch (Section 5.1).

One hosted model reaches parity. `qwen3.8-27b` answered directly at 0.840 against 0.845 with programs, while reporting 523.9 output tokens per direct question against 2.0 for the other hosted models; it reasons before answering, and it exceeded the requested 256 tokens freely. We read this as a boundary of the program advantage. Section 5.6 re-reads it.

5 Direct against program once the conventions are right

5.1 The same programs, executed as labelled

A program names relations, so the archived generations can be executed again under the as-labelled layout on a CPU. The script first replays each run under the layout it was archived with

and stops if anything differs from the archive. As a second check, new GPU runs of the three local models under the as-labelled layout produced programs identical to the archive on 200 of 200 questions each, and the same accuracies. The result is the *as labelled* column of Table 6: every model gains between 0.080 and 0.135, and the five models other than the smallest land between 0.950 and 0.980.

The gain is where the audit predicts (Table 7). For the five other models the two relations that are never true go from 5/11 and 10/16 or 12/16 to all correct, and the axial relations do not change.

		Qwen2.5-1.5B		Qwen2.5-7B		32B and hosted
Relation named	n	published	as labelled	published	as labelled	as labelled
front left	27	6	12	13	23	25
front right	11	2	7	5	11	11
back left	16	9	11	12	16	16
back right	18	8	11	11	16	17
front	10	8	8	10	10	10
back	15	11	11	14	14	14 or 15

Table 7: Questions answered correctly in program mode, by the relation the question names, evaluation split. The last column is common to Qwen2.5-32B and the three hosted models, which differ in this table only in the "back" row (`qwen-3-32b` has 14, the others 15).

The ceiling moves from 0.86 **to** 0.98**.** Under the published layout, 28 questions are wrong in program mode for all six models: 14 anchored on another object, 13 in an ego-relative region, one status question. As labelled, four remain, so the best reachable accuracy goes from 0.860 to 0.980. The four are a count of parked cars to the back right (reference 9, executed 10), a question anchored on "the not standing pedestrian" that the executor refuses, a count of things to the front left of a bus (reference 2, executed 1), and "are there any not standing pedestrians?" (reference *no*, executed *yes*).

5.2 The direct mode, told the convention of the labels

Re-executing programs corrects one mode only. The archived direct answers were given under `direct-1`, which describes sectors the labels do not follow, so the direct mode was rerun with `direct-2`. This prompt states the unsigned angle, says that left and right are not distinguished whatever the name of the relation suggests, gives the four intervals, and says that "front right" and "back left" never contain an object. Table 8 gives the result for the three local models.

	bare answer		working, then answer		program, as labelled	
Model	direct-1	direct-2	direct-1r	direct-2r	free	constrained
Qwen2.5-1.5B	0.505	0.540	0.445	0.345	0.765	0.805
Qwen2.5-7B	0.505	0.545	0.610	0.625	0.950	0.955
Qwen2.5-32B (int4)	0.580	0.640	–	–	0.980	–

Table 8: Evaluation split of the first comparison (200 questions, three drives), ten named classes. `direct-1` states the published sectors, `direct-2` the as-labelled rule; the `r` variants ask for the working. A dash marks a run that was not available when this was written.

Stating the right convention helps the direct mode by 0.035 to 0.060. It helps the program mode by 0.080 to 0.135. The gap between the two modes is therefore larger once both are scored fairly: 0.225, 0.405 and 0.340 for the three sizes, against 0.180, 0.320 and 0.265 under the published reading. With three drives these differences carry no interval; Section 5.5 gives one.

Reasoned direct mode. The bare prompts forbid working, which may handicap the direct mode. The `r` variants ask the model to go through the object list, write down the id of every object the question is about while checking each condition, then count, and to finish with one line "`Answer:`"; only that line is read. The wording (three versions; the first two left the 1.5B model answering in four tokens), the reader's acceptance of a bare one-line answer, and the budget of 2,048 tokens (at 1,536 the 7B model was cut off on 4 to 5 of 100 development questions) were chosen on the development split and only there.

Model	Prompt	Split	accuracy	mean output tokens	no readable answer
Qwen2.5-1.5B	direct-1r	dev	0.360	287	16
Qwen2.5-1.5B	direct-2r	dev	0.370	306	25
Qwen2.5-1.5B	direct-1r	eval	0.445	392	35
Qwen2.5-1.5B	direct-2r	eval	0.345	423	56
Qwen2.5-7B	direct-1r	dev	0.570	382	4
Qwen2.5-7B	direct-2r	dev	0.460	380	4
Qwen2.5-7B	direct-1r	eval	0.610	354	3
Qwen2.5-7B	direct-2r	eval	0.625	427	7

Table 9: Reasoned direct mode in detail, for the runs available. Development has 100 questions and evaluation 200. An answer with no readable final line counts as wrong.

For the 1.5B model, the reasoned mode is below the bare one under both prompts and on both splits: on evaluation 0.445 against 0.505 (`direct-1`) and 0.345 against 0.540 (`direct-2`), with 35 and 56 of 200 responses that never reach a readable answer (Table 9); on development 0.360 against 0.510 and 0.370 against 0.550. For the 7B model the reasoned mode is above the bare one on evaluation under both prompts, 0.610 against 0.505 and 0.625 against 0.545. On development its `direct-2r` score, 0.460, is below its bare `direct-2` score, 0.590 (there is no bare `direct-1` run of the 7B model on development). Under `direct-2` the direction of the effect for the 7B model thus differs between the two splits, and we do not interpret it. Its best reasoned score on evaluation, 0.625, is 0.325 below its program mode at 0.950. For these two models, with this prompt and budget, asking for the working did not close the gap.

5.3 Constrained decoding, a separate condition

All eight invalid programs of the 1.5B model on evaluation emit `"category": "*"` for "things", a token outside the vocabulary, although the prompt lists the ten classes and maps "thing" to `object`. To test the obvious remedy, programs are decoded under a JSON Schema in which categories, statuses, relations and operation names are closed sets, with the explicit refusal still expressible. This changes what the model can emit, so it is reported as its own condition and not folded into the comparison.

Run, as-labelled layout	n	free	constrained	invalid or ill-typed
Qwen2.5-1.5B, development	100	0.800	0.880	$12 \to 0$
Qwen2.5-1.5B, evaluation	200	0.765	0.805	$8 \to 0$
Qwen2.5-7B, evaluation	200	0.950	0.955	$1 \to 0$
Qwen2.5-1.5B, wide scope, flat	280	0.521	0.550	$35 \to 10$
Qwen2.5-1.5B, wide scope, typed	280	0.314	0.354	$170 \to 148$

Table 10: Program accuracy with free and schema-constrained decoding, for the runs available. The last column counts invalid or ill-typed programs, free $\to$ constrained.

On the first comparison, constrained decoding removes every invalid program (Table 10) and is worth 0.040 to the 1.5B model on evaluation and 0.005 to the 7B model, which had one invalid program to lose. On the wide scope of Section 7 it is worth 0.029 to the flat language and 0.040 to the typed one, and it does not remove every invalid or ill-typed program (10 and 148 remain):

the schema closes the vocabulary, not the wiring of types between nodes. It does not make the program say what the question asks.

5.4 Falling back to the direct answer

A deployed system need not choose one mode: it can use the program when it executes and the direct answer otherwise. Computed from the archived rows, with the archived `direct-1` answers as the fallback, this gives 0.835 for Qwen2.5-1.5B as labelled (182 questions answered by the executor, 18 by the direct mode), against 0.765 for the program alone. For the larger models there is almost nothing to fall back on: 0.955 at 7B with three questions passed to the direct mode, and 0.980 to 0.985 for the other four with one or two. Under the published layout the same chain gives 0.755 for the 1.5B model and 0.830 to 0.850 for the others.

5.5 Validation drives, with drive-level intervals

The first comparison has three evaluation drives, and its questions are the ones whose failures revealed the left/right rule. The 1,460 questions of the 146 validation drives are free of both problems, with one reservation: the "things" convention was found on these drives, in the executor check of Table 3, before any model ran on them. Each model was run in both modes under two settings. *Published*: signed sectors, the ten named classes, prompt `direct-1`. *As labelled*: the unsigned rule, every annotation and the ego vehicle as things, prompt `direct-2`. Uncertainty is computed with the drive as the unit: drives are drawn with replacement 10,000 times, every question of a drawn drive is kept, and the statistic is recomputed (Table 11).

Model	Conventions	direct	program	program − direct	drives + / − / =
1.5B	published	0.536 [0.508, 0.566]	0.694 [0.670, 0.718]	+0.158 [+0.122, +0.192]	106 / 25 / 15
1.5B	as labelled	0.582 [0.552, 0.612]	0.747 [0.725, 0.769]	+0.166 [+0.125, +0.205]	108 / 29 / 9
7B	published	0.571 [0.543, 0.599]	0.842 [0.821, 0.863]	+0.271 [+0.244, +0.298]	135 / 3 / 8
7B	as labelled	0.595 [0.566, 0.623]	0.961 [0.950, 0.971]	+0.366 [+0.337, +0.395]	143 / 0 / 3
32B	published	0.628 [0.601, 0.654]	0.853 [0.832, 0.874]	+0.225 [+0.197, +0.253]	122 / 7 / 17
32B	as labelled	0.679 [0.654, 0.704]	0.978 [0.969, 0.986]	+0.299 [+0.274, +0.323]	142 / 0 / 4

Table 11: Qwen2.5 models on 1,460 questions from 146 validation drives never used before; the 32B model is quantised to 4 bits. Brackets are 95% intervals from 10,000 resamples of drives. The last column counts drives where the program mode is ahead, behind, or tied.

The program mode is ahead for all three models under both settings, and no interval on the difference comes near zero. As labelled, the program mode is behind on no drive for the 7B model (ahead on 143) and for the 32B model (ahead on 142). The published rows reproduce the archived picture on new drives (0.694, 0.842 and 0.853, against 0.685, 0.825 and 0.845 on the three mini drives), which indicates that the first comparison was not an accident of its three drives. On the same questions, moving from the published to the as-labelled conventions is worth $+0.053$ $[+0.042, +0.065]$, $+0.119$ $[+0.099, +0.139]$ and $+0.125$ $[+0.105, +0.145]$ to the program mode of the three models in order of size, and $+0.045$ $[+0.027, +0.064]$, $+0.024$ $[+0.005, +0.045]$ and $+0.051$ $[+0.035, +0.068]$ to their direct mode.

		Qwen2.5-1.5B		Qwen2.5-7B		Qwen2.5-32B (int4)	
Family	n	direct	program	direct	program	direct	program
Category only	440	0.695	0.755	0.761	0.995	0.861	0.995
Category with a status	438	0.598	0.970	0.678	0.968	0.751	0.984
Ego-relative region	292	0.486	0.815	0.418	0.938	0.534	0.942
Anchored on another object	290	0.479	0.331	0.393	0.921	0.441	0.979

Table 12: Validation drives, as labelled, by family.

By family (Table 12) the picture is not uniform. For the 1.5B model the program mode is *behind* on questions anchored on another object, 0.331 against 0.479: this is the family where a definite reference must be resolved and where the small model's programs fail most. The 7B and 32B models are ahead in every family, and their direct mode is weakest in the two families that involve a spatial test (0.418 and 0.393 at 7B, 0.534 and 0.441 at 32B).

5.6 The hosted model that appeared to reach parity

Under the as-labelled layout the programs of `qwen3.8-27b` score 0.980, and its archived direct answers stay at 0.840 (Table 6). The parity of the first version, 0.840 against 0.845, came from scoring its programs under the mismatched layout. The new gap of 0.140 is not a clean measurement either. Its direct answers were given under `direct-1`, which told it the published sectors, and it reasoned at length before answering (about 524 output tokens per question). Its direct mode was not rerun under `direct-2`, because it is a hosted endpoint whose alias we cannot pin. Its direct score of 0.840 sits close to the 0.860 that no program could pass under the published layout, which is what a model that applies the stated convention accurately would produce; whether it would approach 0.980 when told the convention of the labels is not known. For this model, neither parity nor a gap is established.

A thinking switch within one checkpoint. The controlled version of that question holds the weights fixed and changes only whether the model may think before answering. Qwen3-8B allows this through its chat template. The design was fixed in advance: prompt `direct-2`, bare answer format, as-labelled layout, and with thinking on a budget of 4,096 tokens for every mode, not tuned. Table 13 lists the configurations completed when this was written.

Thinking	Split	direct	direct tokens	no answer	program	invalid programs
off	dev	0.610	2	0	0.990	1
on	dev	0.630	2,124	37	0.970	2
off	eval	0.580	2	0	0.980	1

Table 13: Qwen3-8B on the first comparison, as labelled. "Direct tokens" is the mean number of output tokens per direct answer, thinking included; "no answer" counts responses cut off by the budget.

Without thinking, this fourth local checkpoint behaves like the Qwen2.5 models: 0.580 directly and 0.980 with programs on evaluation. With thinking, on development, the direct mode is right on all 63 questions it finishes and runs out of its 4,096 tokens on the other 37, which count as wrong; 31 of those 37 are directional questions. So for this checkpoint the direct mode's errors with thinking are a matter of budget, at about 2,100 output tokens per question against a few dozen for a program, and a larger budget might close the gap. The evaluation run with thinking on was interrupted before completing and is not reported.

5.7 The scene-dependence control

Thirteen evaluation questions of the first comparison appear verbatim on two scenes with different reference answers. A system that does not read the objects gives the same answer to both. Under the published reading, the direct mode of Qwen2.5-1.5B gave one answer to a whole group in 4 of 13 groups and was correct across a group once; its program mode never gave one answer to a whole group and was correct across all 13 (on development, 5 groups: direct correct across none, program across 4). The two hosted single-step models were correct across 3 of 13 groups directly and 13 with programs, and `qwen3.8-27b` across 13 in both modes. The re-execution as labelled leaves the 1.5B program mode at 13 of 13. No aggregate accuracy shows this, and it is the kind of control Real-3DQA [8] argues for.

6 Where the program mode fails

The first version triaged the 63 evaluation failures of the 1.5B program mode under the published layout (Table 14). The triage is separable without a model: where the model's program matches the rule grammar's and the answer is still wrong, the program is not at fault.

Failure (published layout)	n	share	cause
Program says something else	27	43%	the model
wrong operation	22		`count` written for an existence question
wrong or dropped status	2		`not with rider` for `without rider`
other	3		
Program right, answer wrong	18	29%	the left/right convention (corrected reading)
Refused at execution	10	16%	ambiguous (8) or missing (2) reference
Malformed output	8	13%	`"*"` as a category
Total	63	100%	

Table 14: The 63 evaluation failures of the Qwen2.5-1.5B program mode under the published layout.

Correction. The first version said that the 18 failures with a correct program were the residual of the diagonal sector boundaries. They are not boundary effects. Executed as labelled, 17 of the 18 become correct; the convention explains them. What is left is almost entirely the small model's own: of its 47 evaluation failures as labelled (0.765), 45 are the three other buckets, which the layout does not change (27 programs that say something else, 10 refusals, 8 malformed outputs).

The largest single failure is a return type. In 22 cases the model writes `count` for a question asking `exists`: "are there any moving cars to the front left of me?" becomes a counting program. The selector is usually right, and the integer it returns can never match a yes/no reference. The eight malformed outputs are the `"*"` category of Section 5.3, which constrained decoding removes.

7 A wider target language

The flat language covers the counting and existence templates. We also built CLEVR's representation [4] adapted to nuScenes: a program is a list of typed nodes in topological order over nineteen functions (`scene`, `filter_category`, `filter_status`, `unique`, `relate`, `relate_ego`, set operations, `count`, `exist`, the queries and comparisons), with types checked before execution. A second manifest built by the same rules covers nine families and 5,830 of the 5,916 mini questions (98.5%, against 27.5% for the flat scope).

Qwen2.5-1.5B, wide scope ($n = 280$)	direct-1	flat program	typed program
Accuracy, published layout (archived)	0.521	0.457	0.296
Accuracy, programs executed as labelled	0.521	0.521	0.314
Answers, published layout	1.000	0.571	0.346
Accuracy when answering, published layout	0.521	0.800	0.856

Table 15: Wide scope. The direct column is the archived `direct-1` answers in both accuracy rows.

The typed language is the most accurate of the three when it produces a program, and it produces one for about a third of the questions (Table 15). Of 280 typed programs, 165 do not type-check: 70 use a constant outside the vocabulary and 85 wire the wrong type into a function,

51 of those being a missing `unique`. The flat language produces 240 valid programs on the same questions and, under the published layout, loses 80 at execution, 42 to ambiguous references and 38 to missing ones. Both languages fail at the same step, resolving a definite reference in a real scene; the flat one fails inside the executor and the typed one at the type check.

The as-labelled layout changes the level and not the conclusion. The flat program mode rises to 0.521, level with the archived direct answers, and the typed one to 0.314. At 1.5B, on the wide scope, the program mode is not ahead of the direct mode, and a more expressive language is harder for the model to write. Chaining the modes (typed program if it executes, else flat program, else direct answer) gives 0.739 as labelled and 0.700 under the published layout, above any single mode. Constrained decoding (Table 10) raises the flat language to 0.550 and the typed one to 0.354.

8 A preliminary study on synthetic scenes

Before the nuScenes protocol existed, the same executor was compared with *trained* baselines on a synthetic corpus under an earlier 45° octant convention, with questions generated from programs. These numbers are not comparable with the sections above.

System	Accuracy	Count MAE	Pair acc.	Params
Majority answer	0.296	–	–	0
Question only (blind)	0.656 ± 0.012	0.722	0.000	35 863
Deep sets	0.655 ± 0.020	0.654	0.022	120 855
Deep sets + counter	0.644 ± 0.018	0.534	0.076	129 176
Symbolic	$\mathbf{0.916 \pm 0.027}$	**0.132**	**0.643**	**0**

Table 16: Synthetic corpus, 45° octants, means over seeds $0, 1, 2$. Pair accuracy requires both members of a contrast pair (one question text, two scenes with different answers) to be correct.

Neither learned model beats the question-only control (Table 16), and on contrast pairs they score 0.022 and 0.076 against 0.643 for the executor. On real nuScenes-QA questions deep sets does not reliably beat the blind prior (margins by seed -0.021, $+0.032$, $+0.025$), and replacing ground-truth objects with simulated detections moves it by $+0.019$, -0.030, $+0.010$. A model whose score is unchanged when the 3D input is corrupted has not been shown to use it.

9 Discussion

What the audit establishes. The released nuScenes-QA answers follow three conventions that the question wording does not suggest. The left/right finding needs no geometry and no executor to verify: 12,392 of 12,392 questions is a property of the question files. Anyone reporting numbers on the benchmark is measuring agreement with these conventions. A system built on the published definition, or on a correct notion of left and right, loses points for it, and a comparison between such a system and one trained on the answers is biased against the first. We make no claim about the generation code.

What the comparison establishes. For these question families, at fixed oracle perception and a fixed frozen model, executing a program the model writes is far more reliable than having the model answer from the object list. On 146 held-out drives the difference is $+0.166$ $[+0.125, +0.205]$ at 1.5B, $+0.366$ $[+0.337, +0.395]$ at 7B and $+0.299$ $[+0.274, +0.323]$ at 32B. Telling the direct mode the right convention moved it by a few points, and asking it to write out its working did not close the gap for the 1.5B and 7B models. Once the convention is right, the program mode of every model but the smallest is between 0.950 and 0.980 on the first comparison, against a ceiling of 0.980.

What our first version got wrong, and why it matters beyond this report. We implemented the published geometry, measured 91 of 100 agreement, read the nine disagreements, and filed eight of them as sector boundary effects. The explanation was plausible and wrong, and it produced a second wrong statement: that the program mode saturates near 0.845. The as-labelled rule was then found by looking at evaluation failures, which is why every check of it reported here uses other data.

What is not established. Nothing here shows that language models cannot reason about space. The direct mode is asked to count rows in a table of several dozen objects, a task small models are known to be poor at, and part of its disadvantage may be context handling and not spatial reasoning. The reasoned-direct ablation covers two models, one prompt and one budget. The one model that reasons at length by default was not rerun under the corrected prompt, and the thinking-switch experiment has its development split only, with 37 of 100 answers cut off by the budget. Perception is an oracle, so the result says nothing about a system that must detect objects first; given Section 3.3, such a system would also be asked to count objects it cannot see. The first comparison has three evaluation drives, and the prompts and layout were developed on v1.0-mini. The 32B model is quantised to 4 bits. On the wide question scope, at 1.5B, the program mode is not ahead.

10 Conclusion

The released nuScenes-QA answers do not distinguish left from right, count the ego vehicle and unnamed classes as things, and count objects without a LiDAR return. We give the rules, evidence on the official train and validation sets, and their limits: they are inferred from answers.

These conventions changed the reading of our own experiment. Under the published definition the program mode appeared to saturate near 0.845 and one reasoning model appeared to match it by answering directly. Under the convention the labels follow, the same programs reach 0.950 to 0.980 for five of six models, the questions no model gets right fall from 28 to 4 of 200, and on 146 held-out drives the 7B and 32B models answer 0.961 and 0.978 of the questions through programs against 0.595 and 0.679 directly. The practical consequence for these question families is to let the language model write the query and let code compute the answer, and to check what a benchmark's labels encode before interpreting a ceiling.

Artifact status. Eight complete archived comparisons, containing 1,580 per-question records, are kept unchanged; the offline `audit_results` command verifies them and their eight re-executed counterparts. The runs of November 2026 are in `results/runs_2026_11`, each with its commit, command, GPU and package versions. Nothing is trained; the 1.5B comparison runs on a T4. Tables 8, 9, 10 and 13 contain the runs complete at the time of writing.

References

[1] S. Yang et al. *LiDAR-LLM: Exploring the Potential of Large Language Models for 3D LiDAR Understanding*, 2023.

[2] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, O. Beijbom. *nuScenes: A Multimodal Dataset for Autonomous Driving.* CVPR, pp. 11618–11628, 2020.

[3] T. Qian, J. Chen, L. Zhuo, Y. Jiao, Y.-G. Jiang. *NuScenes-QA: A Multi-Modal Visual Question Answering Benchmark for Autonomous Driving Scenario.* AAAI, pp. 4542–4550, 2024.

[4] J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. L. Zitnick, R. Girshick. *CLEVR: A Diagnostic Dataset for Compositional Language and Elementary Visual Reasoning.* CVPR, 2017. arXiv:1612.06890.

[5] K. Yi, J. Wu, C. Gan, A. Torralba, P. Kohli, J. B. Tenenbaum. *Neural-Symbolic VQA: Disentangling Reasoning from Vision and Language Understanding.* NeurIPS, 2018. arXiv:1810.02338.

[6] Z. Yuan, J. Ren, C.-M. Feng, H. Zhao, S. Cui, Z. Li. *Visual Programming for Zero-shot Open-Vocabulary 3D Visual Grounding*. CVPR, pp. 20623–20633, 2024. arXiv:2311.15383.

[7] Y. Tian, J. Zhang, Z. Wang, X. Ren, X. Yu, O. Gungor, T. Rosing. *KLDrive: Fine-Grained 3D Scene Reasoning for Autonomous Driving based on Knowledge Graph*. arXiv:2603.21029, March 2026.

[8] X. Ma et al. *Do 3D Large Language Models Really Understand 3D Spatial Relationships?* ICLR, 2026. arXiv:2603.23523.

[9] Qwen Team. *Qwen3-Next-80B-A3B-Instruct model card*. `https://huggingface.co/Qwen/Qwen3-Next-80B-A3B-Instruct`.