LIFT: SCOPED CONSTRAINTS FOR EVALUATING AND TRAINING INSTRUCTION FOLLOWING

Anonymous authors
Paper under double-blind review

ABSTRACT

Common instruction following (IF) benchmarks, such as IFEval, evaluate the ability of language models (LMs) to follow simple constraints that can easily be verified programmatically. These same constraints, in turn, are used as reward signals for reinforcement learning. This design may lead to reward hacking of constraints, pushing models away from the main task. Recent attempts to address this resulted in benchmarks with richer constraints, but required expert labor, remained closed-access, and cannot be easily extended. We present LIFT-Bench, an IF benchmark that asks a model to draft a document given a topic and a structured outline. Constraints are scoped to certain document sections, providing finer control over constraint density. LIFT-Bench includes both verifiable and judged constraints, as well as a training counterpart, LIFT-Train. Importantly, LIFT-Bench provides both main task and instruction-following constraints, mitigating shallow reward hacking. On experiments with multiple models, we find that frontier models struggle with the harder subsets of our benchmark and analyze model performance along multiple axes. We also find that training on LIFT-Train is much less susceptible to reward hacking than training on the common IFTrain. LIFT-Bench and LIFT-Train are publicly available, as well as the data generating pipeline.

1 INTRODUCTION

Instruction following (IF) is the ability to satisfy explicit constraints on a response while still performing the task for which the response is intended. It is typically measured by benchmarks that programmatically check constraint satisfaction in the language model (LM) response (Zhou et al., 2023; Pyatkin et al., 2025). Early work trained for IF with supervised fine-tuning (Wei et al., 2022; Sanh et al., 2022; Mishra et al., 2022) or reinforcement learning from human feedback (RLHF; Ouyang et al., 2022). More recent efforts train LMs to follow instructions by reusing the same programmatic checkers as RL rewards (Pyatkin et al., 2025), a kind of reinforcement learning from verifiable rewards (RLVR; Lambert et al., 2025).

We highlight four shortcomings of existing IF benchmarks. First, the specified constraints, and their checkers, are independent of the main task, leading to potential reward hacking (Amodei et al., 2016; Singhal et al., 2024; Wang et al., 2026). Second, constraint diversity in dominant benchmarks is low, largely consisting of lexical and formatting constraints (Zhou et al., 2023; Pyatkin et al., 2025). Third, constraints in many benchmarks are written by humans (Zhou et al., 2023; Wen et al., 2024; He et al., 2026), which limits size, reproducibility, and adaptation to new domains. In addition, the richest sets are also eval-only or closed (Wen et al., 2024; He et al., 2026). Fourth, difficulty is assessed by total constraint count, ignoring the spread or density of constraints along the response.

We introduce Localized Instruction-Following Tasks (LIFT-Bench), a benchmark of synthetic document-drafting tasks that aims to mitigate these shortcomings. Each LIFT-Bench task provides a topic and an outline—a nested tree of headers whose leaves hold paragraphs, code blocks, tables, and lists—and attaches constraints to individual elements. Each constraint is *scoped*: it is evaluated only against the element it is attached to, not against the whole document. This distinguishes "a paragraph in French somewhere" from "section 2.1 in French". Scoping allows us to define *density*, how many constraints attach to each section. This structured design also renders evaluation more feasible compared to verifying each constraint against the entire document, as in existing benchmarks.

Constraints in LIFT-Bench are either *verifiable*, using deterministic checkers, or *judged*, scored by an LM judge against generated criteria; the latter cover the content a section must contain. Together

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

with scoping, this addresses the above shortcomings: judged content constraints put main-task adherence inside the reward and extend the vocabulary past lexical and formatting checks; synthesis removes the annotator ceiling on size and domain; and density becomes a generation parameter rather than a property the benchmark cannot express.

LIFT-Bench contains 784 examples organized along 10 splits factored by document size, constraint density, and scoping. We also release LIFT-Train, a corresponding dataset for IF training, consisting of 34,258 examples. We evaluated a range of frontier and open models on LIFT-Bench, revealing that models are distinguished by how sharply their scores degrade with increasing density of constraints. The globally-scoped constraints used by current IF benchmark is the hardest regime of all, despite carrying an order of magnitude fewer constraints. On the training side, training on LIFT-Train leads to large improvements on LIFT-Bench, as expected, while training on the common IFTrain (Pyatkin et al., 2025) barely improves on our benchmark and often leads to reward hacking. At the same time, training on LIFT-Train does not degrade on a more challenging out-of-distribution benchmark, AdvancedIF (He et al., 2026), unlike IFTrain.

To summarize, our contributions are:

- LIFT-Bench: a synthetic IF benchmark of document-drafting tasks with scoped constraints, combining verifiable and judged scoring in a single weighted metric.
- LIFT-Train: the training-side release, which is more robust to reward hacking found when training on lexical-only IF rewards.
- An open-source pipeline for synthesizing drafting tasks in `verifiers`-compatible format (PrimeIntellect, 2025), ready for IF-RLVR training and evaluation.
- Analyses that our factored design makes possible: analyses of failures by per-section density and scoping, which require scoped constraints, alongside constraint load, type, and scoring mechanism.

2 Related Work

IFEval (Zhou et al., 2023) was the first benchmark to enable measuring IF, with 541 prompts carrying one to three programmatically checkable constraints. IFBench (Pyatkin et al., 2025) counters overfitting to IFEval with held-out constraint types and publishes IFTrain, a training set that reuses the checkers as rewards. Both benchmarks verify every constraint against the model's entire response (global scope), and in both the main task is not reflected in the metric, leading to reward hacking during RL, as we show later. Pyatkin et al. (2025) also observed such degradation and tried to mitigate it: completions that already satisfy their constraints are rewarded according to reward-model-based classifier. This mechanism tests a response's quality in general, without distinguishing main-task adherence from constraint adherence. LIFT-Bench aims to mitigate these shortcoming.

A second line of work grades responses against LM-judged checklists. InFoBench (Qin et al., 2024) decomposes 500 instructions into 2,250 yes/no requirements, including content requirements; CF-Bench (Zhang et al., 2025) scores 1,000 Chinese prompts against prioritized checklists spanning ten constraint categories; ComplexBench (Wen et al., 2024) scores 1,150 human-authored instructions with yes/no questions (about 4 constraints per instruction), using a mix of rules and GPT-4 (OpenAI, 2023). FollowBench (Jiang et al., 2024) grades difficulty by stacking one constraint per level. These measure richer behavior than the verifiable benchmarks, but are hand-authored and frozen: no training set and no synthesis code, limiting flexibility and modularity.

AdvancedIF (He et al., 2026) (1,645 prompts, 6–10 criteria on average) and ComplexConstraints (Mehta et al., 2026) (75 public prompts, 10–40 criteria each) use expert-written rubrics and show that rubric rewards transfer to held-out IF benchmarks. Both depend on expert authoring, which limits scale and adaptation to new domains, and neither releases its training data; AdvancedIF also keeps its rubric generator, verifier, and RL code closed.

MOSAIC (Purpura et al., 2026) synthesizes 4,000 prompts with up to 20 constraints each. It identifies the important gap between constraint compliance and task success, but attempts to resolve it by modularizing the constraints. Their constraints are limited to "the output's format, style, or structure", and do not test for content inclusion, as we do. Their constraints are globally scoped, and no training set is released. In LIFT-Bench, task adherence is part of the scoring function itself, and structured tasks scope each constraint to one section.

As Table 1 summarizes, LIFT-Bench is the only IF benchmark we know of with content constraints, scoped constraints, and a released training set and pipeline.

Table 1: Comparison with related IF benchmarks. (✓): exists but not publicly released.

	Data		Constraints				Released	
Benchmark	Size	Authoring	Style	Content	Max	Scope	Train set	Pipeline
IFEval	541	manual	✓	✗	3	global	✗	✗
IFBench	300	manual	✓	✗	2	global	✓	✗
ComplexBench	1,150	manual	✓	✓	5	global	✗	✗
AdvancedIF	1,645	manual	✓	✓	10	global	(✓)	(✓)
MOSAIC	4,000	synthetic	✓	✗	20	global	✗	✓
ComplexConstraints	75	manual	✓	✓	40	global	(✓)	✗
LIFT-Bench	784	synthetic	✓	✓	80	section / global	✓	✓

3 LIFT-BENCH

3.1 TASK AND CONSTRAINT MODEL

A LIFT-Bench task consists of a topic, an optional outline, a set of constraints, and a scoring configuration (§3.2). The model is asked to draft a markdown document on the topic; the constraints are what the draft is scored against. See Fig. 1 for an example.

Outline and scope. An outline is a nested tree of headers whose leaves hold content elements: paragraphs, code blocks, tables, and lists. Every constraint is attached to one leaf section and is **scored only against that section's text**, identified by its path in the outline. (A constraint on an artifact—a table, list, or code block—narrows down, to a single artifact within the section.) We call this a *scoped constraint*. Scoping makes each constraint checkable against a passage rather than a whole document, allowing for a fine-grained analysis.

Scoping We separate tasks to *structured* ones, which come with an outline and scoped constraints, and *flat* ones, which do not have an outline and where every constraint scopes globally over the entire response. Flat tasks are the regime existing IF benchmarks occupy, and we include them for comparison under the same pipeline.

Density. Scoping gives an operational definition of *density*: the number of constraints attached to a single section, the smallest unit of text that the model needs to produce. Density is set at generation time (§3.3) rather than inferred after the fact, which lets us vary it against a fixed set of documents.[1]

3.2 SCORING

Every task carries the list of constraints it will be evaluated against, and its score is a weighted sum of per-constraint results in $\{0, 1\}$. Constraints are grouped into **structure** (outline compliance) and **body** (everything else). The relative weight of the two groups is a configuration parameter. In our released configuration, structure receives 10% and content 90%. The same scalar serves as the benchmark metric and as the RL reward (§5).

The LM judge we use is Gemma-4-31b (Gemma Team, 2026); see Appendix B for more details. We ran a human validation study to evaluate its quality, and found it achieved an F1 score of 0.73 against the human annotations; see Appendix C for full details.

Because content constraints carry most of the weight and include per-section content requirements, a response that satisfies every formatting constraint without drafting a meaningful document scores close to zero. This helps avoid reward hacking when training, as we show later (§6).

[1] A flat task has one scope, so its density equals its total constraint count; but that scope is the whole response rather than one section, so flat and structured densities do not sit on a common scale (§3.4).

```
Write a markdown document about 'Ice Hockey World Championships' that has
the following outline:
# Ice Hockey World Championships Overview
## History and Significance
## Format and Qualification Process
## Notable Moments and Records
    [...]
The 'History and Significance' section should follow the constraints
below:
  * The text should explore the significance of the championships in the
    global ice hockey community, discussing how they contribute to [...]
  * The text should detail the origins of the Ice Hockey World
    Championships, tracing back to its first official tournament [...]
  * The text should contain the word 'tournament' at least 3 times.
  * The text should be written in German.
The 'Format and Qualification Process' section should follow the
constraints below:
    [...]
The 'Notable Moments and Records' section should follow the constraints
below:
    [...]
```

Figure 1: Example task in LIFT-Bench, with verifiable (purple) and judge-based constraints (blue).

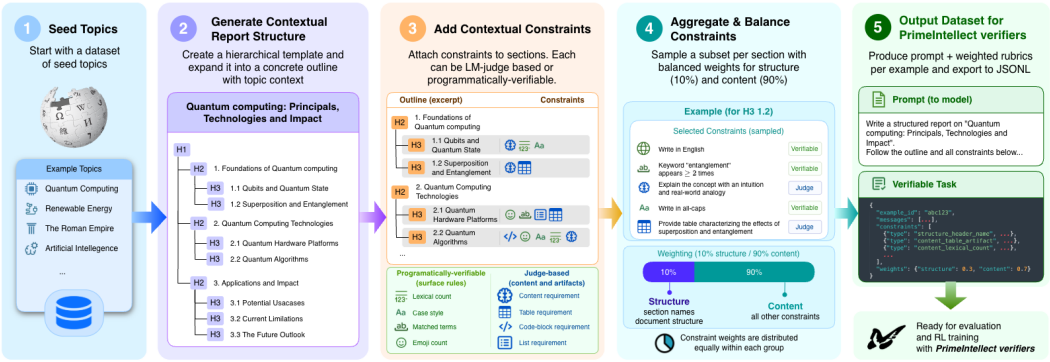


Figure 2: Synthesis pipeline overview.

3.3 Synthesis Pipeline

The pipeline is a sequence of atomic stages, outlined in Fig. 2:[2]

1. **Topic seeding**: sample seed topics from Wikipedia.
2. **Outline construction**: (a) outline templating, where we generate structural document skeletons.; and (b) outline expansion, where an LM fills each skeleton into a concrete, topic-specific outline.
3. **Constraint generation**: parallel per-type generators propose candidate constraints over the outline, one block per constraint type, covering both mechanisms.
4. **Task configuration**
   (a) Constraint sampling: draw the task's constraints from the candidate pool, with knobs for setting how many constraints each section carries, how many sections carry constraints at all, and which constraint types appear.
   (b) Reward shaping: set the structure/content budget split for the task.
5. **Task Scoping**: render the task as a prompt (Fig. 1). Scoping can be **structured**, the outline with its per-header constraints, or **flat**, an IFEval-style prompt in which the same constraints are stated as applying to the entire output. The flat form allows us to compare scoped and globally-scoped prompting directly, holding topic, outline, and constraint content fixed (§4).

[2]Appendix A gives the full description, including prompts, models, and parameters for each released split.

3.4 THE RELEASED SPLITS

Ten splits come from three generation runs: short outlines, long outlines, and flat (no outline). Within each run, every split shares the same topics and outlines; only the number of constraints attached to them changes. Appendix D provides descriptive statistics.

Splits are comparable only if they share the same scoping (flat or structured). A flat split labelled *many* carries far fewer constraints in absolute terms than a structured split with the same label, because a flat task has no sections across which to distribute them—every constraint lands in a single scope. The ten splits are therefore not a monotonic difficulty scale.

4 BENCHMARK EVALUATION EXPERIMENTS

4.1 SETUP

We evaluate 17 models on the 10 released splits. Every example is generated 8 times; an example's score is the mean over its 8 runs, and the intervals are 95% confidence intervals across examples. Judged constraints are scored by Gemma-4-31b (Gemma Team, 2026).

The nine main ones are **(1) closed-weight:** GPT-5.5 (OpenAI, 2026b), GPT-5.6 Sol (OpenAI, 2026c), GPT-5 nano (OpenAI, 2026d), Claude Fable 5 (Anthropic, 2026a) and Claude Opus 4.8 (Anthropic, 2026b); and **(2) open-weight:** DeepSeek (DS) V4-Pro (DeepSeek-AI, 2026), GLM 5.3 (GLM-5 Team, 2026), Kimi K3 (Kimi Team, 2026) and Qwen3.8-2.4T-A95B (Qwen Team, 2026). Appendix E lists the full set of evaluated models.

4.2 OVERALL RESULTS

Table 2 reports per-split scores for a selected 9 models; Appendix F gives the full results for all 17. Structured splits are close to ceiling for the strongest models (GPT-5.5 averages 0.96), but the benchmark is not saturated: no model exceeds 0.85 on any flat split, and weaker models fall to 0.47 on the densest structured splits.

Kimi K3 and DeepSeek V4 Pro have the two highest overall averages: they match the strongest closed models on structured splits and surpass them on flat ones. The two scopes also reorder the ranking: GPT-5.5 is first on structured splits but fifth on flat[3], while Qwen3.8 rises from seventh to third. As all other benchmarks only contain globally scoped constraints, LIFT-Bench provides a novel perspective on model rankings.

Table 2: Mean score per split; best per row in bold; averages are ±95% CI; 'avg (all)' weighs the structured and flat averages equally.

	GPT			Claude		Open Models			
split	5.5	5.6 Sol	5 nano	Fable 5	Opus 4.8	DS V4 Pro	GLM-5.3	Kimi K3	Qwen3.8-A95B
short/few	**0.97**	0.95	0.71	0.92	0.94	0.95	0.87	0.97	0.89
short/some	**0.96**	0.94	0.62	0.91	0.92	0.95	0.88	0.96	0.88
short/many	**0.98**	0.94	0.58	0.82	0.89	0.93	0.83	0.97	0.82
short/numerous	**0.93**	0.92	0.47	0.81	0.82	0.92	0.80	0.92	0.81
long/few	**0.98**	0.96	0.67	0.92	0.96	0.95	0.93	0.96	0.90
long/some	**0.95**	0.95	0.61	0.91	0.91	0.94	0.89	0.95	0.89
long/many	**0.97**	0.93	0.55	0.90	0.92	0.95	0.87	0.95	0.87
long/numerous	0.91	0.89	0.47	0.83	0.87	0.90	0.84	**0.92**	0.80
flat/some	0.71	0.66	0.73	0.64	0.71	0.82	0.60	**0.85**	0.71
flat/many	0.72	0.65	0.68	0.66	0.74	0.80	0.60	**0.83**	0.76
avg (structured)	**0.96**$\pm$0.01	0.94$\pm$0.01	0.59$\pm$0.03	0.88$\pm$0.01	0.90$\pm$0.01	0.94$\pm$0.01	0.86$\pm$0.01	0.95$\pm$0.01	0.86$\pm$0.01
avg (flat)	0.71$\pm$0.03	0.66$\pm$0.03	0.71$\pm$0.04	0.65$\pm$0.03	0.72$\pm$0.03	0.81$\pm$0.03	0.60$\pm$0.03	**0.84**$\pm$0.03	0.73$\pm$0.03
avg (all)	0.84$\pm$0.02	0.80$\pm$0.01	0.65$\pm$0.02	0.76$\pm$0.02	0.81$\pm$0.02	0.87$\pm$0.02	0.73$\pm$0.01	**0.90**$\pm$0.02	0.79$\pm$0.02

4.3 WHERE MODELS FAIL: LOAD, MECHANISM, AND TYPE

Intuitively, the more constraints are added, the more difficult it is to follow the instruction. To investigate this, we compare two models: the top frontier model GPT-5.5 and the weaker GPT 5

[3]Constraint count does not explain the drop of closed models; see Appendix G.

nano. We break down performance by the number of judged versus verified constraints. LIFT-Bench uniquely supports this analysis thanks to its factored design. As Fig. 3 shows, GPT-5.5 declines mildly across the entire plane, while the weaker nano model collapses, and does so more steeply along the judged axis. In other words, the number of judged constraints affects performance more than the number of verifiable constraints, in weaker models.

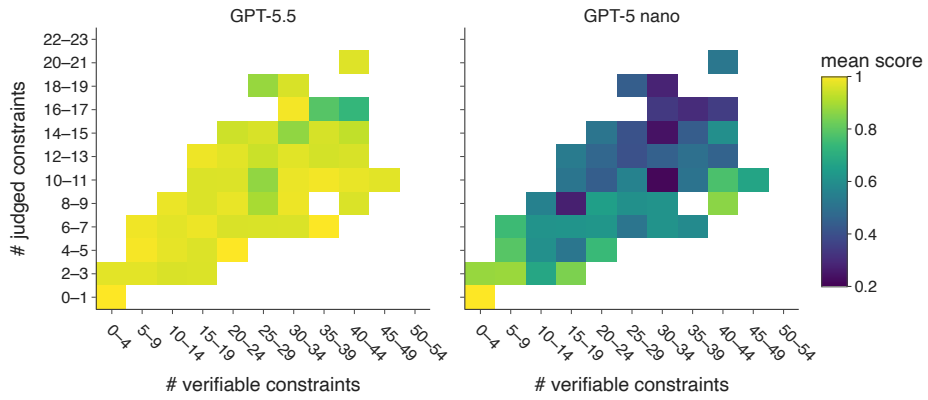


Figure 3: Benchmark scores broken by number of judged and verifiable constraints. In the lightest cell (bottom left) GPT-5.5 and GPT-5 nano both score near 1, but at the densest loads the weaker model falls to 0.2–0.5 while the stronger stays above 0.7.

Next, we break down the results by constraint density, that is, the number of constraints per section. To do so, we fix a section density and increase the total load, by adding constraints elsewhere in the document. As Table 3 shows, frontier models are only mildly affected by this intervention (from near zero for Kimi K3 to -1.5% per density step for GPT-5.5), while smaller ones degrade steadily (-5.5% for GPT-5 nano), indicating that their ability to satisfy a given local constraint is affected by more remote constraints. The results over all evaluated models are given in Appendix H.[4]

Table 3: Change in section score, estimated via linear regression and averaged over density types. At fixed per-section density, frontier models are only mildly affected by constraints elsewhere in the document while smaller models continue to degrade. The full results are in Table 11 and the fitting procedure in Appendix I.

	% score change by document size		
Model	short	long	mean
Kimi K3	-0.9%	$+0.2\%$	-0.4%
Claude Opus 4.8	-0.4%	-0.5%	-0.4%
GPT-5.6 Sol	-0.4%	-0.9%	-0.7%
GPT-5.5	-1.3%	-1.7%	-1.5%
GPT-5 nano	-7.0%	-4.1%	-5.5%

Finally, we analyze how models follow different types of instructions (Fig. 4). Some types are near ceiling for every frontier model (`doc_structure`, `emoji_count`, `content`), while artifact constraints (`list`, `tables`, `code_snippets`) and `case_type` vary most. GPT-5 nano trails the frontier models on every type, most on `tables` and `case_type`; `list` and `tables` are where mid-tier models such as Claude Sonnet 4.6 fall behind (Appendix H).

The frontier models agree on most types but not all: on `code_snippets` Kimi K3 scores 0.98 against 0.73–0.83 for the other three. What the breakdown shows is a shape rather than a ranking: models at a similar average score can fail at different things. Full grids are in Appendix H.

[4]Note that the globally-scoped regime is a different case. There, a flat task means that every constraints belongs in a single global scope, which the LM must satisfy. This explains why the frontier models' robustness to the rest of the document does not help them on flat tasks.

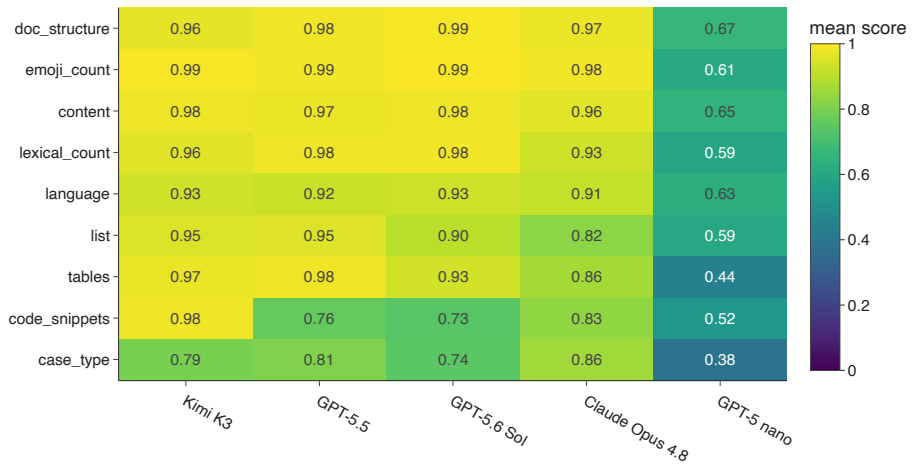


Figure 4: Benchmark scores per constraint type. Models display distinct constraint-type fingerprints, even where their averages are close.

Load and type breakdowns are also possible on MOSAIC (Purpura et al., 2026). Per-section density and scoping are not, because existing IF benchmarks check every constraint against the whole response; in LIFT-Bench both are generation parameters rather than incidental properties of the prompts.

5 LIFT-TRAIN

LIFT-Train is the training-side release of the same pipeline: training and test tasks are produced by one generator, and the benchmark's scoring function (§3.2) is used unchanged as the RL reward. Accordingly, most of the reward consists of content constraints, so an output that satisfies every formatting constraint without drafting a meaningful document earns almost nothing.

In contrast, in IFTrain (Pyatkin et al., 2025) constraints are lexical and verifiable but do not test whether the response's content meets the request. Pyatkin et al. (2025) acknowledge the potentially-hackable reward and add a general-purpose classifier, based on a reward model, on top of the verifiable reward, bonusing or penalizing rollouts by the classifier's output. However, that signal rates overall quality without distinguishing satisfaction of the verifiable constraints from adherence to the requested task. In LIFT-Train, task adherence is scored by constraints of its own, inside the same reward, rather than by a separate quality signal. Thus, we hypothesize that LIFT-Train is less likely to be prone to reward hacking. We perform multiple training runs on various configurations from LIFT-Train, as well as comparing with IFTrain, to isolate which aspects are more important.

5.1 TRAINING MIXTURES AND SETUP

Training mixtures. Using the factored splits of §3.4, we vary the training mixture along two independent axes: training data composition and curriculum.

Composition We consider four data compositions: *Structured-only* only contrains structured data splits; *Flat-only* only contains flat splits (closest to IFEval-style data); *Combined* contains both structured and flat data (controlled for size); and *Combined + IFBench constraints* adds IFBench-style constraint types to the pool.

Curriculum Each composition is trained either at *fixed difficulty*, sampling across splits without regard to load, or on a *curriculum* that presents splits from easy to hard.

As a control we train on IFTrain (lexical, verifiable-only constraints). The flat-only LIFT-Train mixture is the closer comparison: it matches IFTrain's prompt format and constraint style, and differs in that its reward includes a content component.[5]

Training setup. We use GRPO (Shao et al., 2024) on three base models: Qwen2.5 7B and 32B (non-reasoning) (Yang et al., 2024) and Qwen3-32B (reasoning) (Yang et al., 2025) for a total of 180 steps; the Qwen2.5 models continue to step 390 to look for long-training pathologies.

6 Training Experiments

6.1 Main training results

Table 4 compares the results of training on our LIFT-Train with training on the common IFTrain (Pyatkin et al., 2025). As expected, training on IFTrain performs well on IFEval and IFBench, as IFTrain was designed to mimic their constraints. Similarly, training on LIFT-Train performs well on LIFT-Bench. However, considering more out-of-distribution benchmarks, AdvancedIF and ComplexBench (§2), training on IFTrain degrades performance while LIFT-Train provides small improvements. This suggests that IFTrain leads to more reward hacking than LIFT-Train.

Table 4: Benchmark scores after GRPO training on Qwen2.5-7B (step 180), with the change in points against the base model in parentheses. Shaded cells are in-distribution; other cells are out-of-distribution. Each recipe gains most on the benchmark its reward can see, but only LIFT-Train also improves the external judged benchmarks.

Benchmark	base	IFTrain	LIFT-Train
IFEval	0.712	0.764 (+**5.2**)	0.723 (+1.1)
IFBench	0.270	0.430 (+**16.0**)	0.278 (+0.8)
LIFT-Bench (10-split avg)	0.423	0.451 (+2.8)	0.822 (+**39.9**)
AdvancedIF (3-split avg)	0.392	0.374 (−1.8)	0.401 (+0.9)
ComplexBench (EN)	0.324	0.316 (−0.8)	0.327 (+0.3)

To probe this, we examined the responses by Qwen2.5-7B over the course of training on IFTrain. By step 180, the reward saturates at 1.0 while median visible length on the held-out validation prompts plummets from 114 to 28.5 words, and to 18 by step 390 (Fig. 9a). Some of the cases we observed in the GRPO rollouts demonstrate reward hacking: by step 180, bare constraint tokens with high rewards; by step 390 we see empty section headers, long repetitions of a single phrase, and other pathological behaviors. See Appendix J for some example rollouts as well as more quantitative analysis. In contrast, the same base model trained on LIFT-Train does not exhibit this high reward-hacking tendency during training.

We attribute this to the content-adherence component in LIFT-Train. However, one potential confound with this comparison is that our benchmark differs from IFTrain in more than the reward structure (different format, structure, and topics). To control for this, we compared also with training on a flat-only version of LIFT-Train. It matches IFTrain's surface form (flat prompts, IFEval-style constraints), but its reward includes content adherence. We found this set to produce healthy training, without length pathologies as those mentioned above. While the comparison to IFTrain is not perfect (prompts still cover different topics), this is another evidence that content adherence mitigates reward hacking.

6.2 Mixture composition and training order

Next, we examine the role of the training set composition and curriculum, training Qwen3-32B on each mixture (Fig. 5). The mixtures seem largely interchangeable when inspecting the overall scores. However, when breaking down performance to different splits in LIFT-Bench, we do see some composition effects: training on a single scoping also improves the other, but falls short of

[5]One caveat for this comparison is that prompt content still differs: our topics are wiki-seeded drafting tasks, while IFTrain's are taken from WildChat (Zhao et al., 2024).

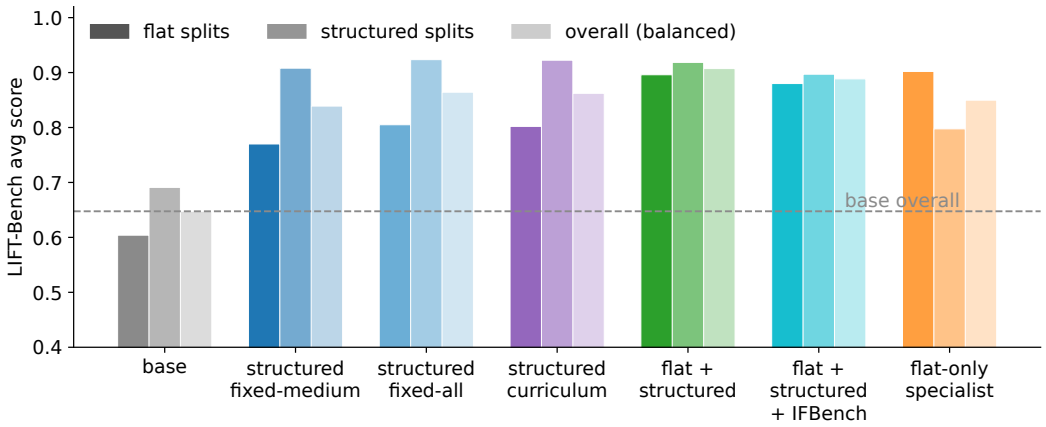


Figure 5: Comparison of training mixtures on Qwen3-32B: LIFT-Bench score at GRPO step 180, averaged over the flat splits, the structured splits, and their macro-average (mean of the flat and structured averages).

the combined mixture, which does well on both: structured-only training (fixed-all) reaches 0.805 on the flat splits against 0.896 for the combined mixture, and flat-only training reaches 0.798 on the structured splits against 0.919 (base: 0.604 flat, 0.691 structured).

Training curriculum of easy-to-hard splits also makes no difference: the structured curriculum run scores on par with its fixed-difficulty counterpart (Fig. 5). This may be due to the split design (§3.4): adjacent splits overlap, so ordering does not add that much diversity. We leave deeper exploration of training curriculum to future work.

7 Conclusion

Instruction-following is measured by checking constraints and trained by rewarding the same checks, and both halves of that arrangement are blind to whether the requested task was performed. LIFT-Bench closes the blindspot by scoping constraints to individual sections and scoring content requirements alongside formatting ones, which makes constraint density a controlled variable and main-task adherence part of the reward rather than an assumption about it. Our experiments with various closed frontier models and leading open models demonstrate that LIFT-Bench is a useful benchmark for evaluating instruction following. Its modular composition exposes weaknesses of different models along constraint density, mechanism, and type. In addition, the released training counterpart, LIFT-Train, is less susceptible to reward hacking than the common IFTrain, thanks to the content-adhering part of its reward.

There are a few possible extensions for LIFT-Bench to pursue. A finetuned rubric judge and a learned rubric generator, in the style of AdvancedIF's verifier (He et al., 2026), would harden the scoring stage that the benchmark and the reward both rely on, and can be easily added to the pipeline. The task distribution can grow along further axes: richer constraint compositions, adding the conditional and chained dependencies of ComplexBench (Wen et al., 2024) to our conjunctive ones, and multi-turn splits, where an editorial follow-up tests ongoing edits rather than drafting alone.

The benchmark and its synthesis pipeline will both be made publicly available upon acceptance.

AI USE STATEMENT

In this work, we used generative AI tools for assistance in several aspects:

Dataset generation. The judge-based constraints in our data-synthesis pipeline are all LLM-based. Also, for structured tasks, the draft's structure is also generated by an LLM.

Data presentation. We used generative AI tools to edit figures and tables for better readability.

Code generation. We used generative AI tools for assistance in adding a generic task prompt formatting mechanism to the data-generation pipeline (similar to other existing mechanisms in our source code) and in open-sourcing our code.

Editorial suggestions. We used generative AI tools to suggest editorial changes for terms or phrasings for better readability, as well as drafting some initial paragraphs. Almost all text has been rewritten or edited manually and all of it has been manually vetted.

We have reviewed all AI-assisted work. We checked LLM-generated plots for potential inconsistencies with the actual data, LLM-generated code was verified and tested for correctness by 1 author, LLM-based editorial suggestions were reviewed by at least 2 authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

We do not foresee any noteworthy ethical problems with this paper.

REPRODUCIBILITY STATEMENT

The code and data associated with this paper will be made publicly available upon acceptance. The appendix contains implementation details needed for reproducing our work, when used with the released code base.

REFERENCES

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

Anthropic. System card: Claude Haiku 4.5. `https://www-cdn.anthropic.com/7aad69bf12627d42234e01ee7c36305dc2f6a970.pdf`, 2025.

Anthropic. System card: Claude Fable 5 and Claude Mythos 5. `https://anthropic.com/claude-fable-5-mythos-5-system-card`, 2026a.

Anthropic. System card: Claude Opus 4.8. `https://anthropic.com/claude-opus-4-8-system-card`, 2026b.

Anthropic. System card: Claude Opus 5. `https://anthropic.com/claude-opus-5-system-card`, 2026c.

Anthropic. System card: Claude Sonnet 4.6. `https://anthropic.com/claude-sonnet-4-6-system-card`, 2026d.

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026. Describes DeepSeek-V4-Pro (1.6T parameters, 49B activated) and DeepSeek-V4-Flash.

Gemma Team. Gemma 4 technical report, 2026. URL `https://arxiv.org/abs/2607.02770`.

GLM-5 Team. GLM-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026. GLM-5.3 shares the GLM-5.2 base model; all gains come from post-training. Weights and release notes at `https://github.com/zai-org/GLM-5`.

Yun He, Wenzhe Li, Hejia Zhang, Songlin Li, Karishma Mandyam, Sopan Khosla, Yuanhao Xiong, Nanshu Wang, Xiaoliang Peng, Beibin Li, Shengjie Bi, Shishir G. Patil, Qi Qi, Shengyu Feng, Julian Katz-Samuels, Richard Yuanzhe Pang, Sujan Gonugondla, Hunter Lang, Yue Yu, Yundi Qian, Maryam Fazel-Zarandi, Licheng Yu, Amine Benhalloum, Hany Awadalla, and Manaal Faruqui. Advancedif: Rubric-based benchmarking and reinforcement learning for advancing llm instruction following. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2026. arXiv:2511.10507. Introduces the AdvancedIF benchmark and the RIFL rubric-reward training pipeline.

Yuxin Jiang, Yufei Wang, Xingshan Zeng, Wanjun Zhong, Liangyou Li, Fei Mi, Lifeng Shang, Xin Jiang, Qun Liu, and Wei Wang. Followbench: A multi-level fine-grained constraints following benchmark for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024. arXiv:2310.20410.

Kimi Team. Kimi K3: Open frontier intelligence. *arXiv preprint arXiv:2607.24653*, 2026.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James Validad Miranda, Alisa Liu, Nouha Dziri, Xinxi Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Christopher Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tülu 3: Pushing frontiers in open language model post-training. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=i1uGbfHHpH.

Sushant Mehta, Liudas Panavas, Suhaas Garre, and Edwin Chen. ComplexConstraints and beyond: Expert rubrics for RLVR. *arXiv preprint arXiv:2606.09118*, 2026.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. Cross-task generalization via natural language crowdsourcing instructions. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3470–3487, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.244. URL https://aclanthology.org/2022.acl-long.244/.

OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

OpenAI. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

OpenAI. Introducing GPT-5.4 mini and nano. https://openai.com/index/introducing-gpt-5-4-mini-and-nano/, 2026a.

OpenAI. GPT-5.5 system card. https://openai.com/index/gpt-5-5-system-card/, 2026b.

OpenAI. GPT-5.6 system card. https://deploymentsafety.openai.com/gpt-5-6, 2026c. Covers the GPT-5.6 Luna, Terra and Sol variants.

OpenAI. OpenAI GPT-5 system card. *arXiv preprint arXiv:2601.03267*, 2026d. Safeguards for the API-only gpt-5-mini and gpt-5-nano variants are given in the addendum on OpenAI's Deployment Safety Hub.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 27730–27744. Curran Associates, Inc., 2022. doi: 10.52202/068431-2011. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/b1efde53be364a73914f58805a001731-Paper-Conference.pdf.

PrimeIntellect. Verifiers: Environments for llm reinforcement learning. https://github.com/PrimeIntellect-ai/verifiers, 2025. Software framework for defining verifiable-reward RL environments/tasks.

Alberto Purpura, Li Wang, Sahil Badyal, Eugenio Beaufrand, and Adam Faulkner. Deconstructing instruction-following: A new benchmark for granular evaluation of large language model instruction compliance abilities. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1334–1349, 2026.

Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hannaneh Hajishirzi. Generalizing verifiable instruction following. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2507.02833. Introduces the IFBench benchmark and IFTrain RLVR data.

Yiwei Qin, Kaiqiang Song, Yebowen Hu, Wenlin Yao, Sangwoo Cho, Xiaoyang Wang, Xuansheng Wu, Fei Liu, Pengfei Liu, and Dong Yu. InFoBench: Evaluating instruction following ability in large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, 2024.

Qwen Team. Qwen3.8-Max: A new bar for coding and cowork. https://qwen.ai/blog?id=qwen3.8, August 2026. Model card for the open-weight base, Qwen3.8-2.4T-A95B: https://huggingface.co/Qwen/Qwen3.8-2.4T-A95B.

Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M Rush. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=9Vrb9D0WI4.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Prasann Singhal, Tanya Goyal, Jiacheng Xu, and Greg Durrett. A long way to go: Investigating length correlations in RLHF. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=G8LaO1P0xv.

Xiaohua Wang, Muzhao Tian, Yuqi Zeng, Zisu Huang, Jiakang Yuan, Bowen Chen, Jingwen Xu, Mingbo Zhou, Wenhao Liu, Muling Wu, et al. Reward hacking in the era of large models: Mechanisms, emergent misalignment, challenges. *arXiv preprint arXiv:2604.13602*, 2026.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=gEZrGCozdqR.

Bosi Wen, Pei Ke, Xiaotao Gu, Lindong Wu, Hao Huang, Jinfeng Zhou, Wenchuang Li, Binxin Hu, Wendy Gao, Jiaxin Xu, Yiming Liu, Jie Tang, Hongning Wang, and Minlie Huang. Benchmarking complex instruction-following with multiple constraints composition. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, volume 37, 2024. arXiv:2407.03978.

Wikipedia contributors. Wikipedia:Vital articles/Level 5. https://en.wikipedia.org/wiki/Wikipedia:Vital_articles/Level_5, 2023. We use a historical snapshot of this list, containing approximately 40,000 unique topics.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Tao Zhang, Chenglin Zhu, Yanjun Shen, Wenjing Luo, Yan Zhang, Hao Liang, Tao Zhang, Fan Yang, Mingan Lin, Yujing Qiao, Weipeng Chen, Bin Cui, Wentao Zhang, and Zenan Zhou. CF-Bench: A comprehensive constraints-following benchmark for LLMs. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 32926–32944, 2025.

Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. Wildchat: 1m chatgpt interaction logs in the wild. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Bl8u7ZRlbM.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023. Introduces the IFEval benchmark and the strict/loose accuracy metrics.