From Few to Many: Self-Improving Many-Shot Reasoners Through Iterative Optimization and Generation

Xingchen Wan, Han Zhou, Ruoxi Sun, Hootan Nakhost, Ke Jiang, Sercan Ö. Arık

{xingchenw, soarik}@google.com

Paper: https://arxiv.org/abs/2502.00330

Google Research

Many-shot ICL (MS-ICL)

Increased context length in modern models has enabled *many-shot* learning

- 4-8 examples -> 50-100 examples (or more)

Previous works show performance benefits from scaling examples.

- **Reinforced ICL setup:** assume presence of inputs and final labels; **any intermediate outputs are model-generated.**
- Retains the {question, intermediate output, answer} of the *correctly* predicted examples.

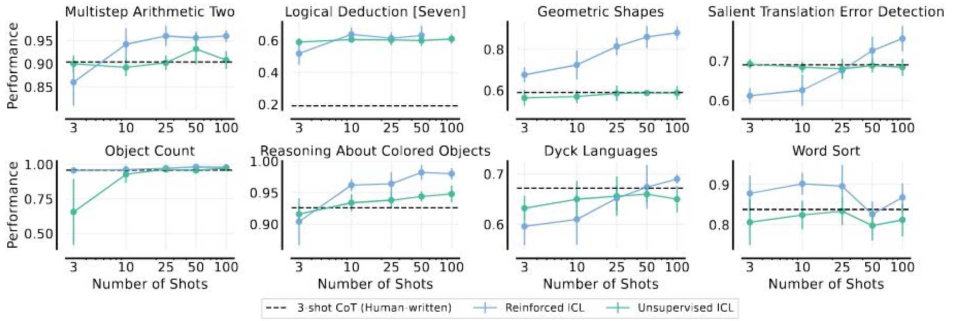
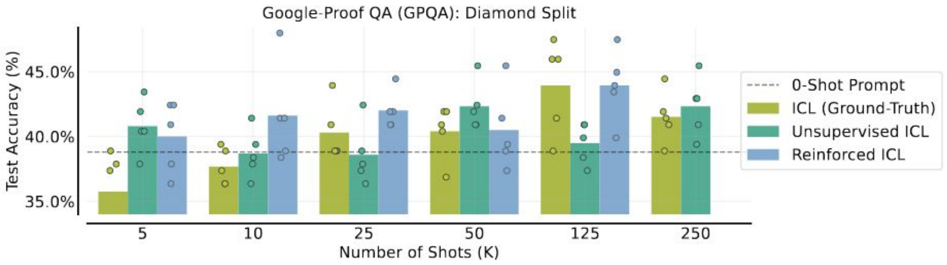


Figure 9 | **BIG-Bench Hard.** Reinforced and Unsupervised ICL with varying number of shots, averaged across five random seeds. We evaluate test performance on a held-out set of 100 problems. The error bars denote standard deviation. Reinforced ICL outperforms Unsupervised ICL for all tasks, which in turns outperforms the human-written chain-of-thought (CoT) prompt. Averaged across tasks, CoT prompting using human-written rationales gets a success rate of 72.1%, Unsupervised ICL obtains **77.1%**, while Reinforced ICL gets **83%**.



Source: arXiv:2404.11018

Google Research

What drives many-shot performance improvements?

Is scaling **inherently beneficial** or because scaling **increases the chance of choosing good examples**?

Can we do better than naive scaling / selection?

- Scaling right to the context limit is often not feasible (due to latency concerns) which necessitates some kind of selection;
- Existing works usually select randomly

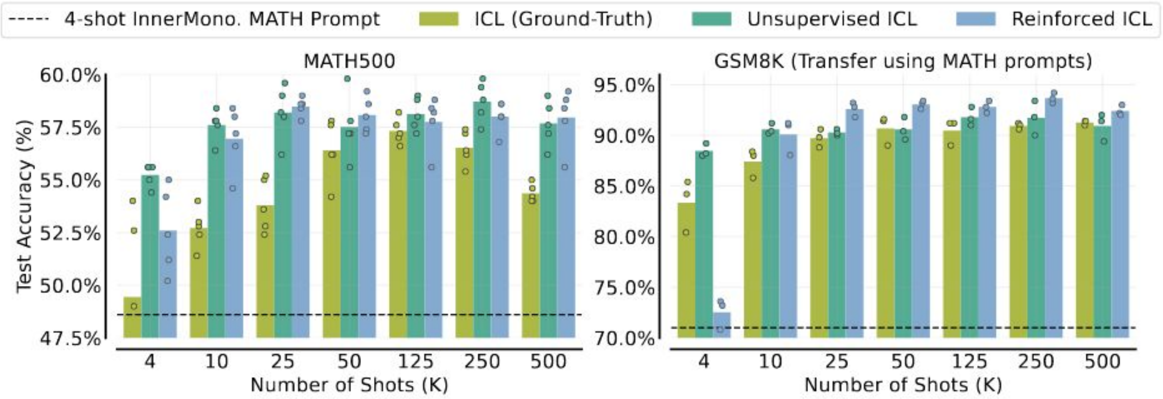


Figure 7 | **Many-shot Reinforced and Unsupervised ICL for problem-solving** generally outperform ICL with ground-truth MATH solutions. **MATH**. (Left) The bar plots depict the average performance across five random seeds on the MATH500 test set. Each random seed (denoted by the dots) corresponds to a different subset of problems along with ground truth or model-generated solutions (if any) in the prompt. **Transfer to GSM8K**. (Right) We see that the prompt obtained from MATH transfers well to the GSM8K test split containing 500 problems. Our results with many-shot ICL outperform the 4-shot Minerva prompt, which obtains a test accuracy of 55.7% on MATH500 and 90.6% on GSM8K.

Source: arXiv:2404.11018

Google Research

Analysis: Attributing performance to individual examples

Attribution answers the question in the previous slide:

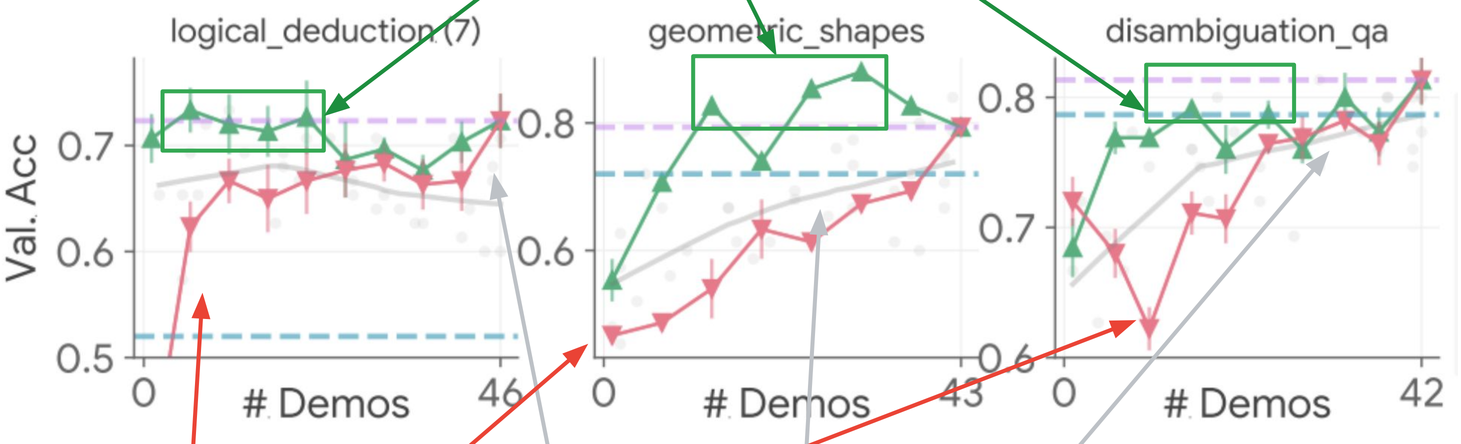
- If *many-shot* performance is driven by *few examples*, we should in principle be capable of tracing it back to few examples.
- Using these few examples alone should lead to comparable / better performance to *all* examples.

Attribution Step:

- *Binary vector representation*: if we have |E| demos in total, any subset can be represented as a |E|-dimensional binary vector:
    - [1, 1, ..., 1] -> all demos
    - [1, 0, ..., 0] -> first demo only
- *Gaussian process modelling:* we can sample N such demo subset, evaluate their validation performance and build a GP that learns the function: demo subset -> performance.
- *Gradient-based attribution*: We can compute the gradient w.r.t. each dimension of the binary vector and average across all examples to approximate *saliency*.
- *Ranking:* We then rank the examples based on their imputed saliency from the previous step.

Analysis Results

Acc. vs. # demos if we select the examples with most positive examples first



Acc. vs. # demos if we select the examples with the negative examples first

Overall trendline (LOWESS fitted)

Key takeaways: Selection > scaling; many-shot performance can still be (disproportionately) driven by few high-performing examples.

Analysis: Can we still benefit from scaling examples?

Can “many-shot” still help?

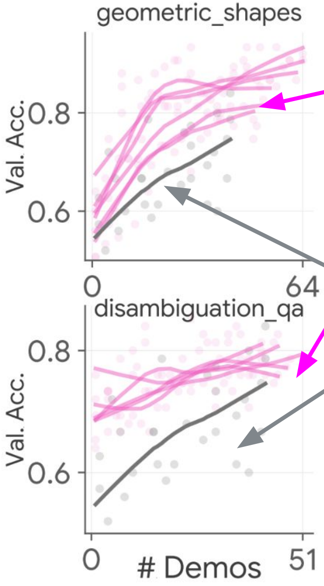
- If improvements from MS-ICL can disproportionately come from few-shot, do we still need many-shot?

Answer: Yes!

Key idea:

1. Identify the high-performing demonstrations (with interpretable ML or pruning)
2. Use this subset as “seed” demonstrations to **re-generate** new examples (reinforced ICL setup)
3. Step 1 and 2 can be **iterated until convergence.**

Note magenta line both **ends later (larger pool)** and is higher than gray lines at **any # demos**

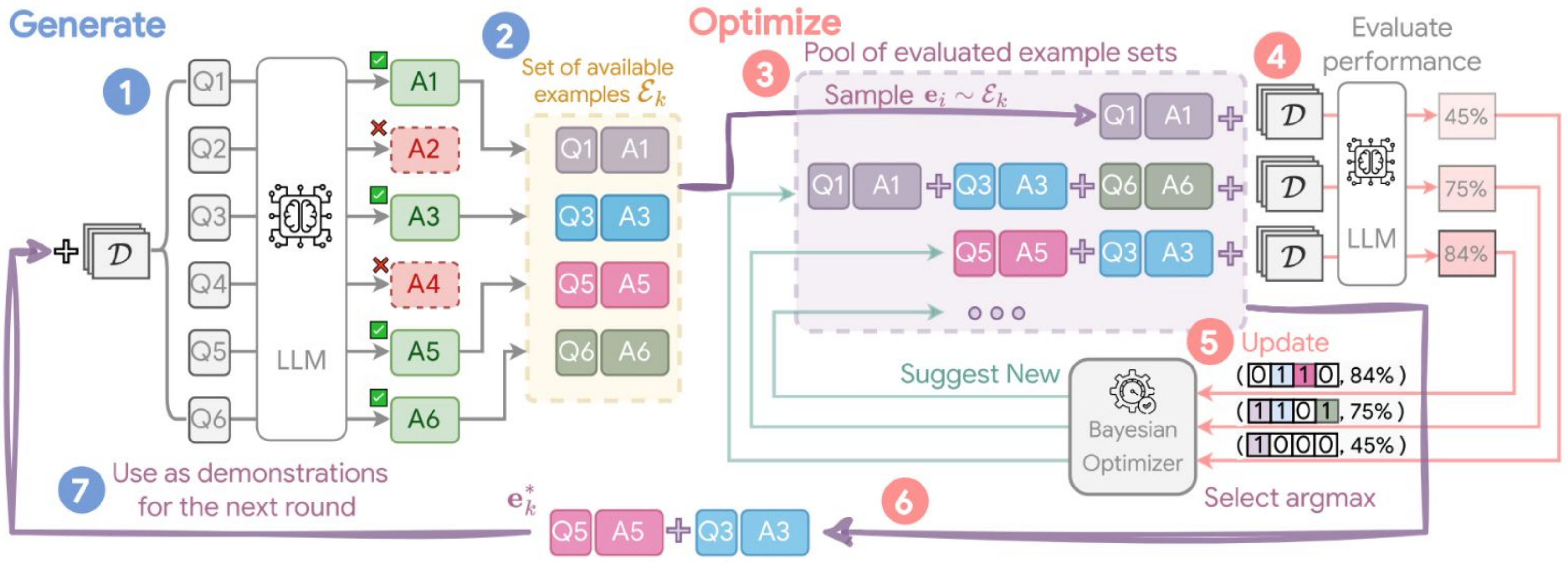


Reuse top-5 subsets and regenerate demonstrati ons

Original trendlines

Method

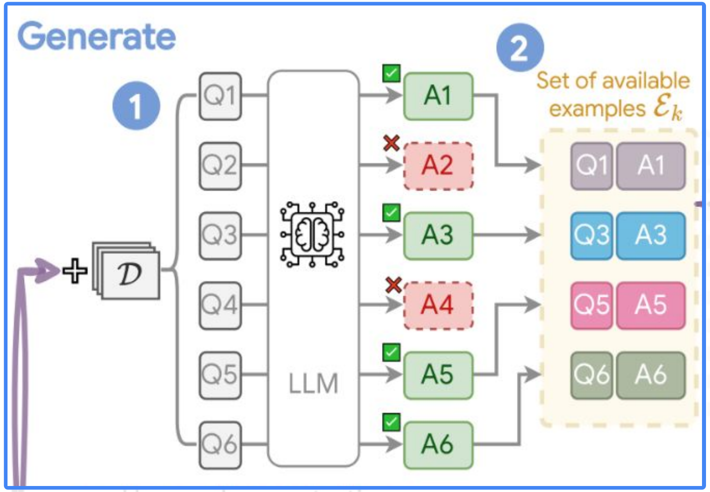
We propose **BRIDGE** to algorithmically take advantage of the findings in the previous slides



Google Research

Method

We propose **BRIDGE** to algorithmically take advantage of the findings in the previous slides



Generate Step

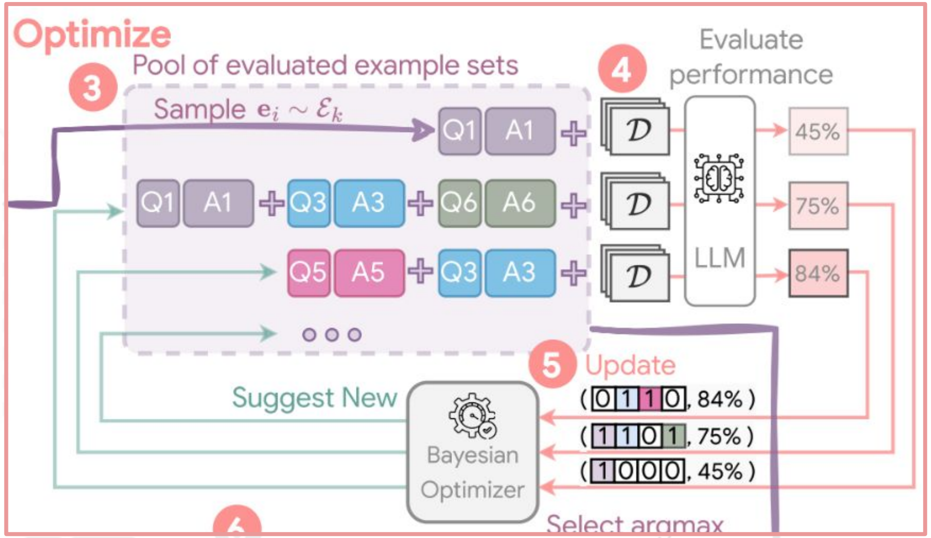
- *Reinforced ICL* / STaR-like setup.
- Use the LLM to predict on the labeled dataset, and retain the reasoning traces / intermediate outputs that led to correct answers.
- For the very first iteration, either use *zero-shot* (using model's inherent capability) or any *human-annotated* examples.

Method

We propose **BRIDGE** to algorithmically take advantage of the findings in the previous slides

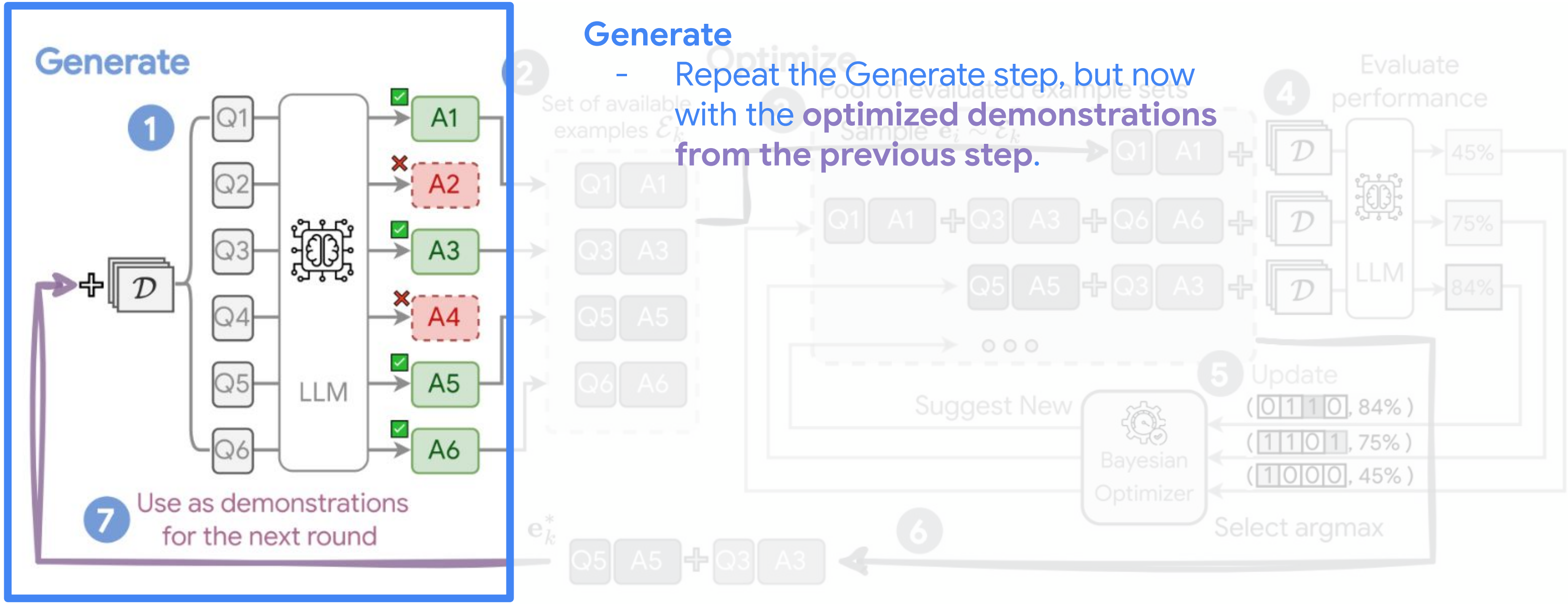
Optimize Step
From the examples generated from the previous step, run optimization loop to figure out the *important demos.*

- **Reuse** the Gaussian Process surrogate for combinatorial Bayesian optimization (learn and optimize the binary vector -> score)
- **Reuse** the labeled dataset used for generate demos for validation



Method

We propose **BRIDGE** to algorithmically take advantage of the findings in the previous slides



Generate

- Repeat the Generate step, but now with the **optimized demonstrations from the previous step.**

Experiments

Using the entire labeled set
Direct: Q + Final A
CoT: Q + reasoning + Final A

Iterated Reinforced ICL
Reuse all correctly predicted examples as demos
Our algorithm without the Optimize step.

Tasks	All		Reinf. ICL	Iterative Reinf.		BRIDGE (*Ours*)				
	Direct	CoT								
# Iterations	-	0	0	1	2	1o	1G	2o	2G	3o
causal_judgement	$61.0_{4.7}$	$62.7_{2.1}$	$66.3_{4.8}$	$68.7_{1.9}$	$69.3_{2.7}$	$68.3_{1.5}$	$62.7_{1.6}$	$59.7_{1.5}$	$\mathbf{72.0}_{0.0}$	$70.0_{2.0}$
date_understanding	$87.2_{2.0}$	$86.0_{2.3}$	$88.8_{2.5}$	$93.0_{1.0}$	$94.9_{1.3}$	$92.2_{1.5}$	$\mathbf{97.0}_{0.7}$	$94.8_{1.9}$	$95.0_{1.2}$	$95.5_{1.8}$
disambiguation_qa	$74.2_{2.2}$	$63.3_{1.1}$	$76.8_{2.4}$	$74.6_{1.4}$	$75.1_{1.5}$	$71.8_{2.4}$	$77.5_{3.6}$	$80.5_{1.8}$	$\mathbf{81.3}_{2.9}$	$78.8_{1.5}$
dyck_languages	$16.8_{2.9}$	$39.0_{3.7}$	$55.5_{3.6}$	$64.4_{5.3}$	$74.4_{3.6}$	$49.2_{2.7}$	$76.2_{3.8}$	$\mathbf{80.0}_{2.7}$	$77.5_{1.1}$	$76.8_{3.8}$
formal_fallacies	$82.8_{3.7}$	$86.8_{1.3}$	$86.2_{1.1}$	$88.1_{0.9}$	$89.4_{1.4}$	$86.0_{2.1}$	$85.0_{2.5}$	$\mathbf{90.8}_{2.3}$	$90.8_{2.8}$	$88.2_{2.3}$
geometric_shapes	$69.0_{4.1}$	$61.8_{4.2}$	$80.2_{2.8}$	$81.0_{2.5}$	$82.3_{1.7}$	$78.5_{2.1}$	$82.5_{3.6}$	$89.2_{3.8}$	$\mathbf{92.3}_{1.1}$	$89.2_{0.8}$
hyperbaton	$70.8_{4.1}$	$93.2_{3.1}$	$90.2_{1.1}$	$91.5_{2.2}$	$86.2_{2.5}$	$96.5_{0.9}$	$94.2_{1.5}$	$94.8_{2.8}$	$96.5_{0.5}$	$\mathbf{97.2}_{0.4}$
logical_deduction (7)	$56.8_{4.4}$	$63.0_{7.4}$	$65.8_{3.5}$	$68.9_{2.6}$	$69.5_{2.9}$	$70.2_{1.5}$	$70.8_{4.5}$	$\mathbf{71.7}_{3.7}$	$71.5_{1.8}$	$69.2_{2.2}$
movie_recommendation	$\mathbf{75.0}_{1.0}$	$63.7_{2.2}$	$65.2_{1.6}$	$68.8_{2.0}$	$67.0_{4.3}$	$67.0_{1.2}$	$69.5_{0.5}$	$69.3_{3.1}$	$72.8_{1.8}$	$67.0_{1.2}$
multistep_arithmetic_two	$86.5_{2.2}$	$96.8_{0.8}$	$96.5_{0.5}$	$95.9_{0.8}$	$94.5_{1.3}$	$96.2_{0.8}$	$94.5_{1.1}$	$97.0_{0.7}$	$\mathbf{98.0}_{0.7}$	$96.8_{1.8}$
object_counting	$92.5_{2.3}$	$84.8_{4.3}$	$95.5_{0.9}$	$95.8_{2.2}$	$95.1_{1.6}$	$\mathbf{96.2}_{0.4}$	$96.0_{1.9}$	$94.5_{1.1}$	$94.2_{0.4}$	$95.0_{0.7}$
ruin_names	$85.2_{3.1}$	$85.5_{2.1}$	$89.8_{1.9}$	$88.6_{1.5}$	$90.5_{0.9}$	$\mathbf{90.8}_{1.1}$	$88.8_{1.7}$	$89.2_{1.5}$	$88.8_{2.4}$	$90.3_{0.8}$
salient_translation_error_detection	$66.0_{2.4}$	$56.2_{1.5}$	$69.0_{1.6}$	$73.8_{1.1}$	$73.4_{1.3}$	$68.8_{0.8}$	$71.0_{0.7}$	$69.5_{2.2}$	$74.0_{0.7}$	$\mathbf{74.5}_{1.1}$
snarks	$94.1_{1.8}$	$95.5_{2.3}$	$92.7_{3.2}$	$94.3_{1.9}$	$95.5_{1.5}$	$93.4_{3.0}$	$95.8_{0.0}$	$95.1_{1.6}$	$96.9_{1.5}$	$\mathbf{97.6}_{1.8}$
sports_understanding	$93.8_{1.3}$	$94.2_{1.3}$	$93.0_{1.4}$	$94.1_{0.9}$	$95.4_{1.2}$	$92.8_{1.9}$	$\mathbf{97.0}_{1.2}$	$96.2_{0.8}$	$95.8_{0.4}$	$95.8_{0.8}$
tracking_shuffled_objects (7)	$76.0_{7.2}$	$52.5_{2.1}$	$62.3_{4.2}$	$64.5_{2.2}$	$65.5_{4.6}$	$95.8_{0.4}$	$95.0_{1.2}$	$\mathbf{100.0}_{0.0}$	$97.0_{0.7}$	$99.5_{0.5}$
Average	74.22	74.06	79.61	81.61	82.37	82.11	84.61	85.77	**87.13**	86.33

BIG-Bench Hard*, gemini-1.5-pro-001

Reinforced ICL
CoT: Q + reasoning + Final A, *but only if Final A is correct*
Significant gain!

Ours
Significant gain over reinforced ICL!
- Each generate and optimize step led to improvement until 3o
- Hypothesis: At 3o, there is no / few “bad demos” and pruning led to regression

Google Research

Experiments

Tasks	Reinf. ICL	Iterative Reinf.		BRIDGE (Ours)				
# Iterations	0	1	2	1o	1G	2o	2G	3o
Hendryck's MATH	63.80	63.60	64.60	63.00	60.20	64.00	<u>64.80</u>	**65.40**
GSM-Hard	69.71	70.15	68.90	<u>72.92</u>	72.03	71.58	**73.91**	71.58

Method	Exec. Acc.	Breakdown S	M	C
Direct	57.7	64.0	49.4	44.1
CHASE prompt	60.1	67.2	51.9	40.7
CHASE + BRIDGE				
Round 0	59.1	65.7	51.3	42.1
Round 1	61.2	68.6	50.6	48.3
Round 2	<u>62.0</u>	68.5	53.0	49.0
PEFT (LoRA)				
$n_{\text{train}} = 256$	58.2	64.0	52.2	40.7
$n_{\text{train}} = 1024$	60.2	66.6	53.0	42.1
$n_{\text{train}} = 4096$	61.3	67.5	53.9	46.2
$n_{\text{train}} = 9428$ (All)	**63.8**	68.6	58.8	48.9

Table 4 | Execution accuracy on the BIRD dev set with `gemini-1.5-pro-001`. {S, M, C} refer to the accuracy aggregated across {Simple, Moderate, Challenging}-level problems based on assigned difficulty.

A (simplified) variant of CHASE-SQL prompt (divide-and-conquer + CoT) + 4 handcrafted demos

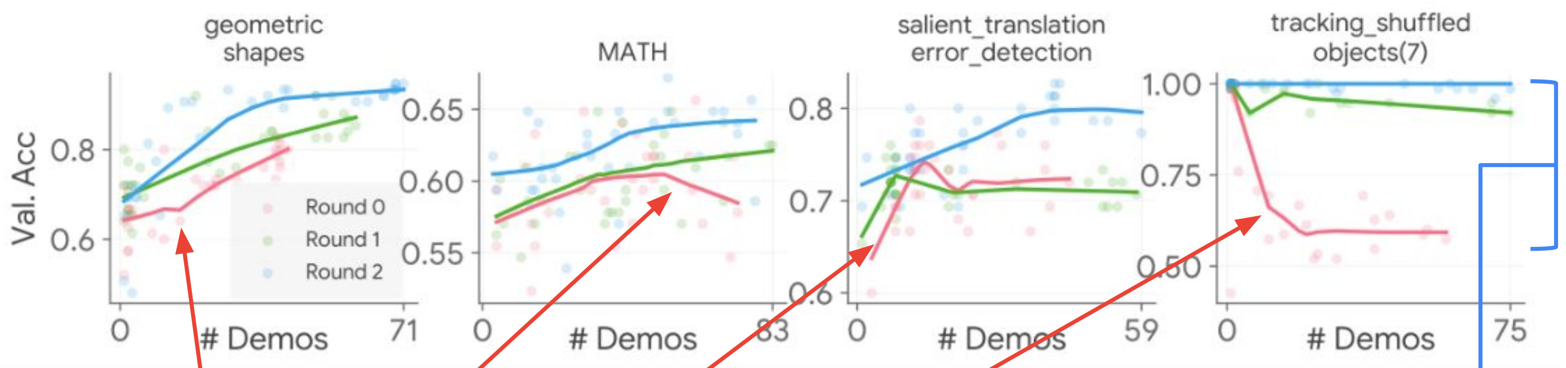
3 round of BRIDGE with 256 labeled examples + zero-variance filtering (generated demos appended to handcrafted demos)

LoRA finetuning with many more training examples

- Prompt design + many shot can perform on par or better than PEFT with much higher data requirement!

Google Research

Analysis



The benefit of naive scaling is task-dependent
Ranges from strong positive correlation (left) to increase then plateau (middle two) to outright negative correlation (right)

BRIDGE alleviates the task-specificity and each round (mostly) incrementally improved over the previous round

Google Research

Conclusion

Analyzing and Improving Many-shot Performance

1. Analyze what drives many-shot performance
2. Propose BRIDGE to automate **optimization** and **generation** to iteratively improve many-shot performance

Google Research