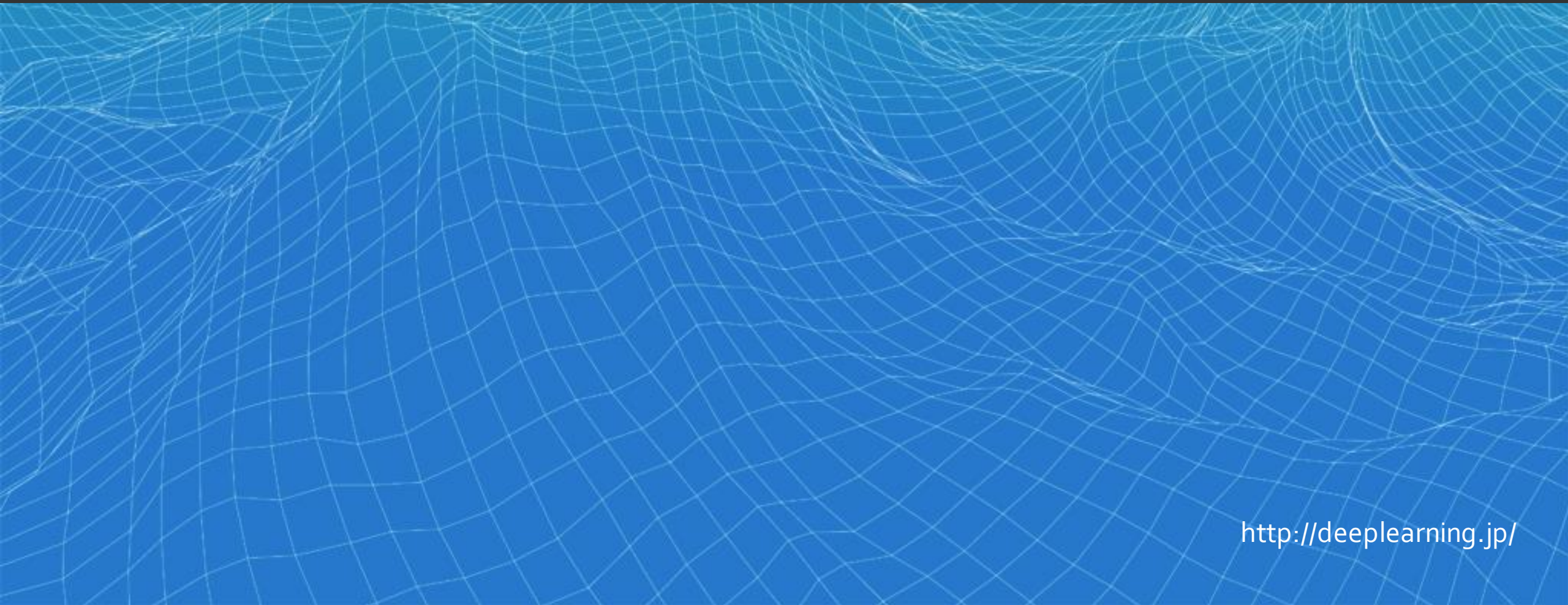
DEEP LEARNING JP
[DL Papers]

The Surprising Effectiveness of Test-Time Training for Abstract Reasoning

2024.11.21 Kexin Song Matsuo-Iwasawa Lab M2



http://deeplearning.jp/

Information

- **The Surprising Effectiveness of Test-Time Training for Abstract Reasoning**
  - arXiv: https://arxiv.org/abs/2411.07279
  - Github: https://github.com/ekinakyurek/marc?tab=readme-ov-file

- Author：Ekin Akyürek Mehul Damani Linlu Qiu Han Guo Yoon Kim Jacob Andreas

  (Massachusetts Institute of Technology)

1. Abstract

Researchers used **Test-Time Training (TTT)** to temporarily update model parameters during reasoning with loss functions derived from input data. They validated TTT's effectiveness in enhancing LLM reasoning on the ARC(Abstraction and Reasoning Corpus) benchmark, analyzed key components for applying TTT, and proposed two innovations: **TTT Data Generation** and a **self-consistency component**. Results showed that models with TTT rivaled or outperformed symbolic reasoning models on ARC.

2. Test-Time Training

- First introduced in 2020 for visual models by UC Berkeley and UCSD: https://arxiv.org/abs/1909.13231
- Core Mechanism:
  - **Dynamic parameter updates** during inference using explicit gradient steps. Allowing it to handle unseen or shifted data distributions more effectively.
- Operates under **low-data conditions**:
  - **Unsupervised learning** with single inputs.
  - **Supervised learning** with one or two labeled examples.

The general TTT process

1. Start with pre-trained model parameters θ_0.
2. Generate a small training dataset D_{TTT} from the test data.
3. Update parameters θ_d by minimizing a loss function $L(\ D_{TTT}\ ; \theta)$.
4. Use the updated parameters for predictions.
5. Reset the parameters to the original state θ_0 after each test input or batch.

Key Design Choices in TTT

- Challenges:
  - TTT's design space is vast, but effective strategies for new task learning remain unclear.
  - Need to understand interactions with **pretraining** and **sampling strategies**.
- Contributions:
  - Systematic study of TTT design choices and their effects.
  - Identification of critical components for few-shot learning:
    - **Initial fine-tuning** on similar synthetic tasks.
    - **Enhanced leave-one-out task generation.**
    - Training **per-instance adapters**.
    - **Self-consistency** under reversible transformations.

ARC Challenge

1. ARC Challenge Overview:

- The Abstraction and Reasoning Corpus (ARC) tests abstract reasoning abilities through solving visual puzzles.
- Each task consists of 2D grids with shapes or patterns involving up to 10 colors, created using shared transformation rules $y = f(x)$.
- The goal is to predict y^{test} for x^{test} by reasoning about transformations.

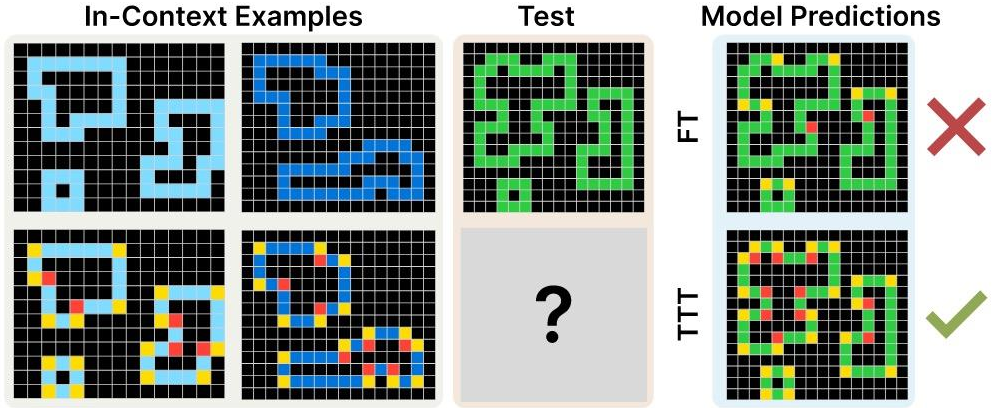
2. Two Approaches:

- Program synthesis methods: Discover the transformation function f and apply it to test examples.
- Fully neural methods: Predict outputs y^{test} without explicitly modeling f . **(used in this paper)**

3. Method in this Study:

- A formatting function (denoted str) converts 2D grids into string representations for input to LMs.

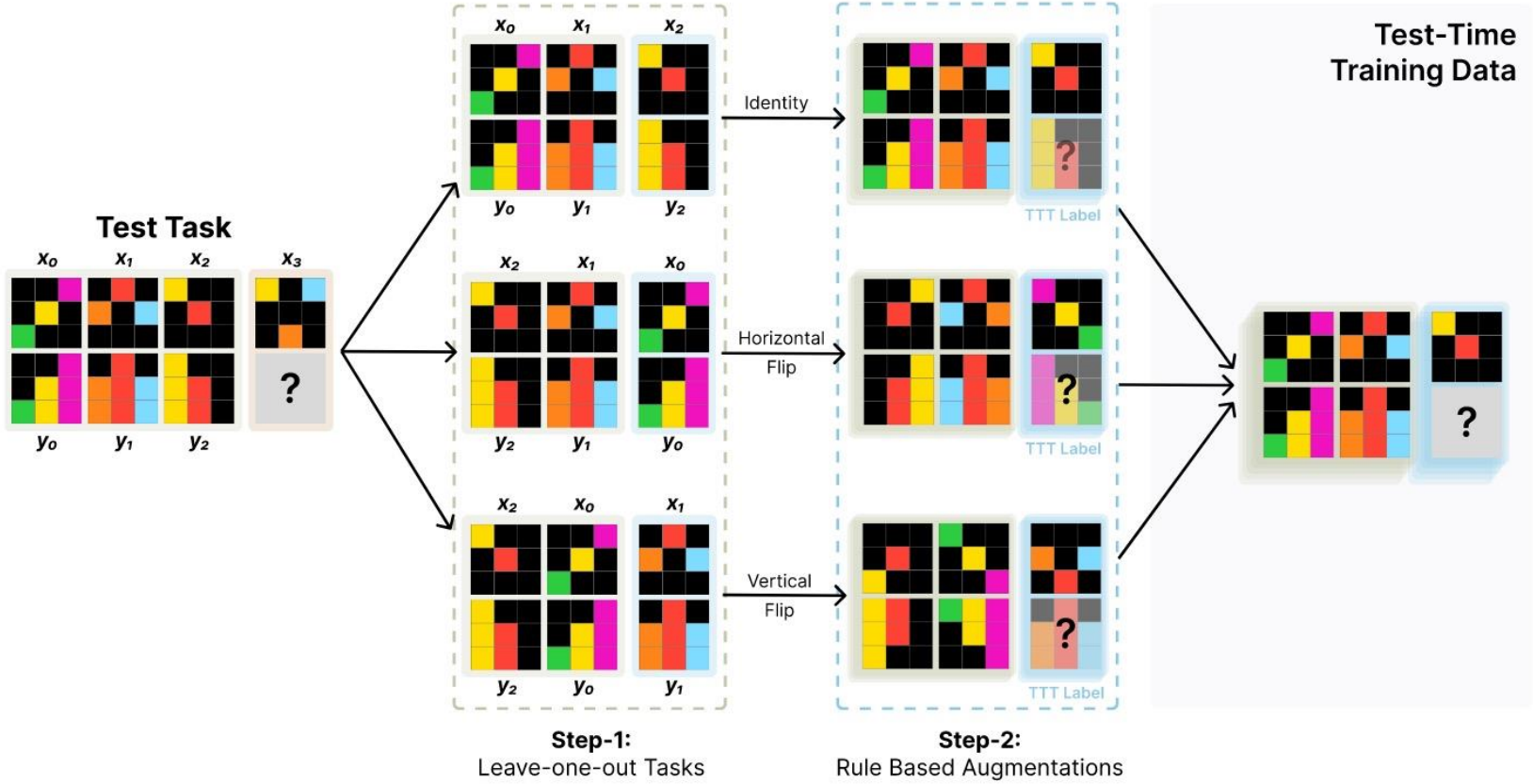
Example of the task in ARC:



3.1 Data Generation for TTT

Two-step process:

- **Leave-one-out tasks:** Exclude one pair as the test example, and use the remaining pairs as training examples.
- **Rule-based augmentations:** Apply reversible transformations like rotation, flipping, scaling, etc.



Comparison with E2E Learning:

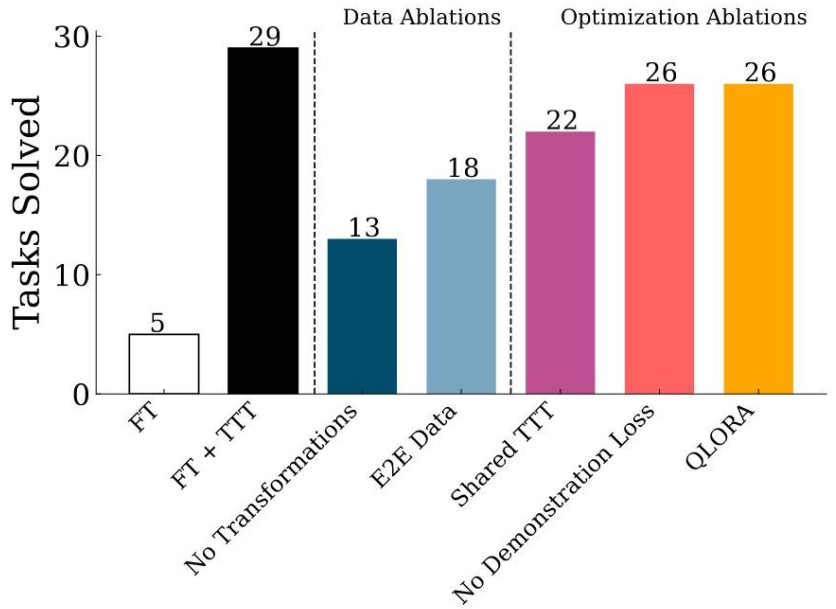
- **End to End Approach:**
  - Treats input-output pairs independently as supervised examples.
  - Does not maintain in-context demonstrations.
  - More computationally efficient.

3.2 Optimization Objectives in TTT

- **LoRA Optimization:**
  - Task-specific parameter updates while freezing most base model weights.
  - Efficient computation with retained general model capabilities.

Results and Impact of TTT

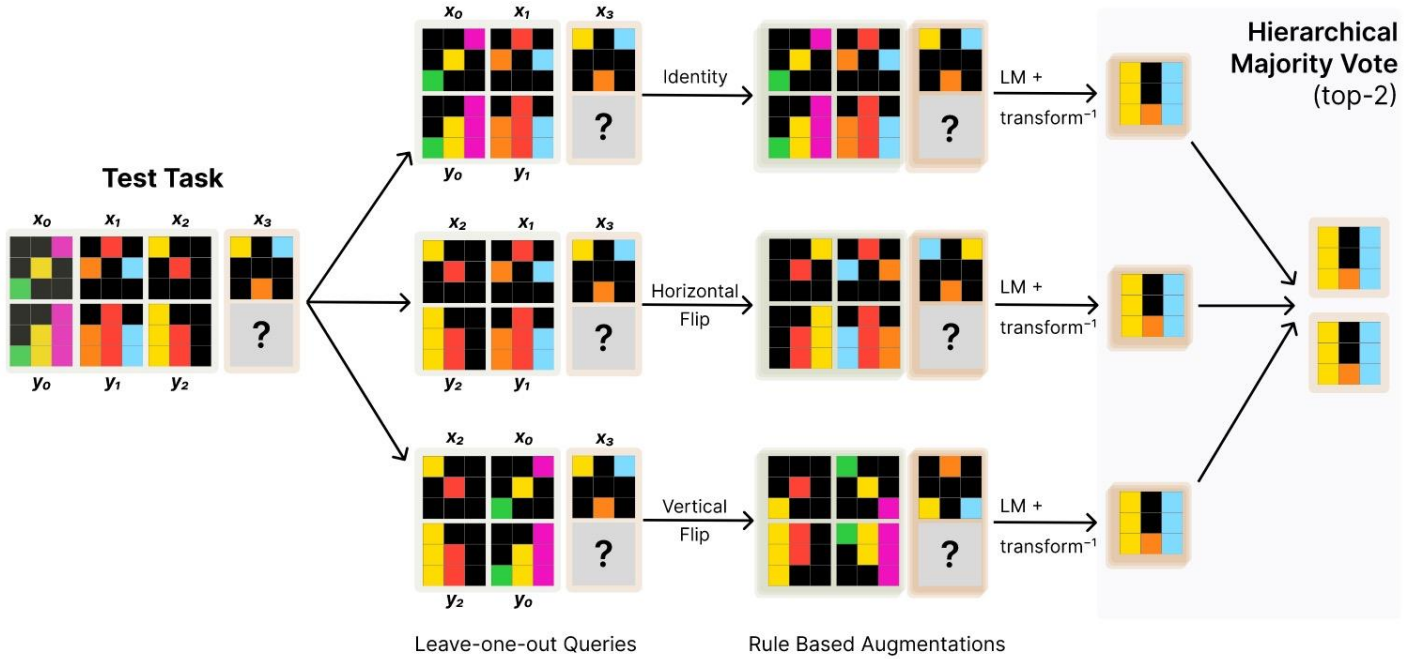
- **TTT Accuracy Boost:**
  - Accuracy increased sixfold (from **5 to 29**).
- **In-Context Learning (ICL):**
  - Outperformed end-to-end tasks.
  - E2E approach showed a **38% performance drop** under identical conditions.



4.1 Inference Enhancements

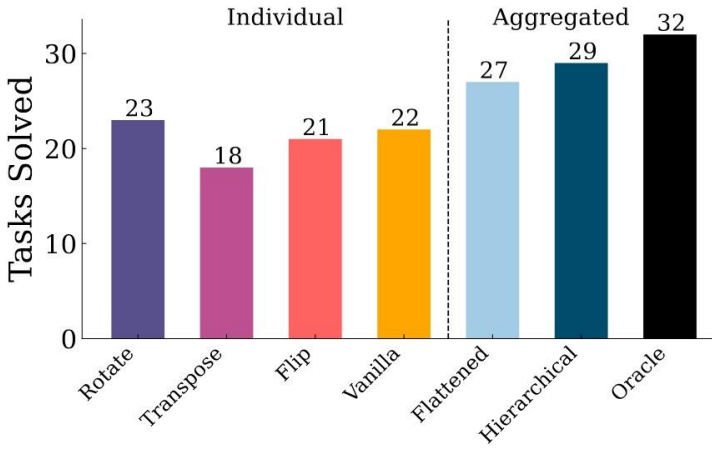
- **Challenges:** No chain-of-thought (CoT) in ARC, making majority voting inapplicable.
- **Augmented inference strategy:**
  - Use transformations to generate diverse predictions.
  - Apply **hierarchical voting** to select the best predictions.

– **Advantages:** Reduces bias in handling demonstration sequences.



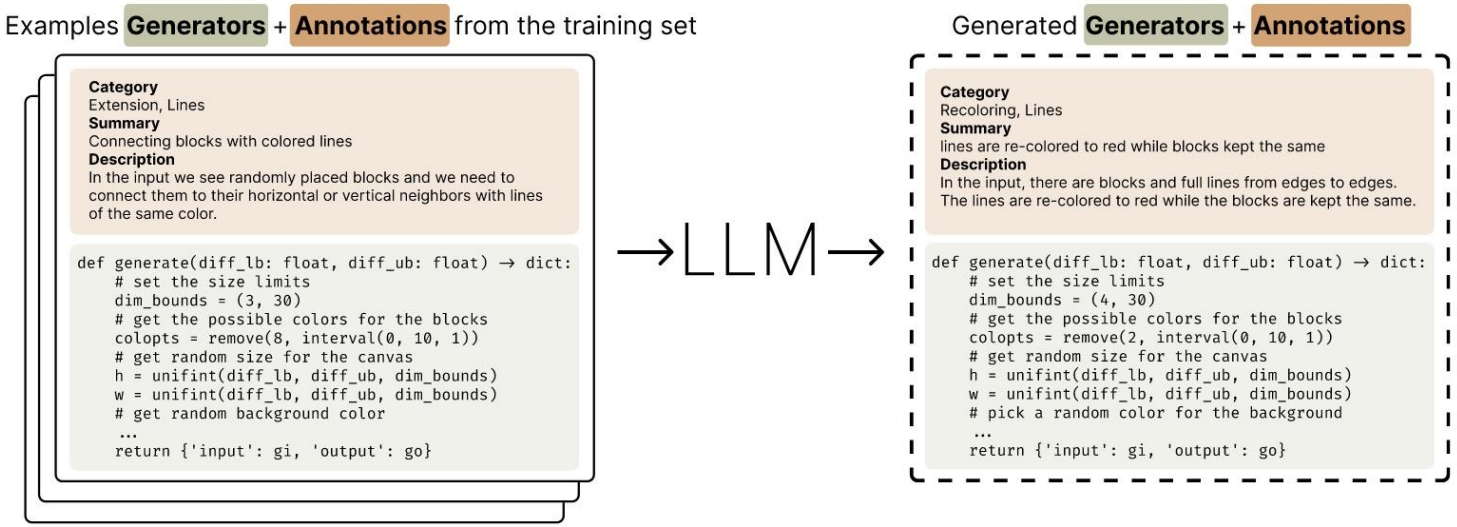
4.2 Integrated Prediction: Voting Strategy

- **Two-Stage Voting Process:**
  1. **Intra Transformation Voting:**
     - Group predictions by transformation type t.
     - Select top 3 predictions with the highest frequency in each group.
     - Supplement with row-majority and column-majority predictions if necessary.
  2. **Global Voting:**
     - Combine candidates from intra-transformation voting.
     - Select the top 2 overall predictions.
     - Prioritize **identity transformation** predictions in case of ties.
- **Results:**
  - **Self-consistency voting** enhances accuracy and aligns with prior findings.
  - **Hierarchical voting** outperforms **flattened voting** for accuracy improvement.



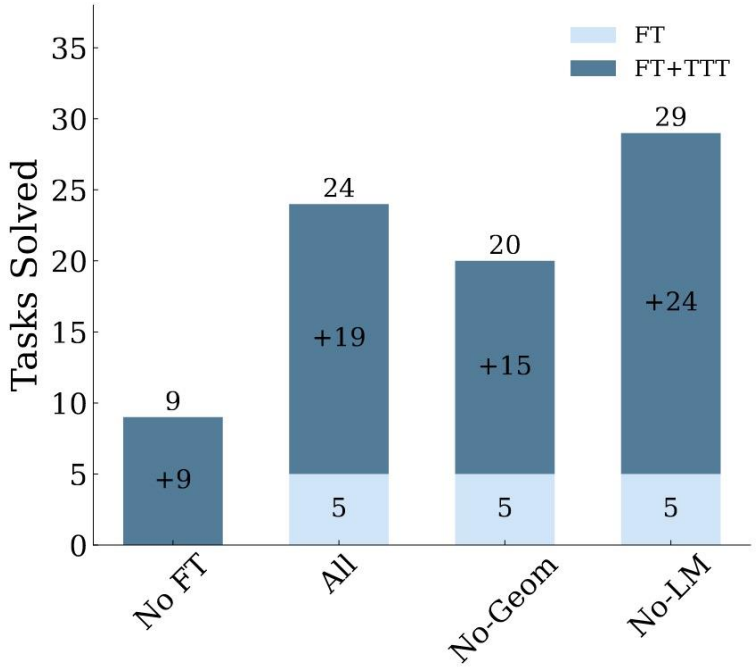
5.1 Data Preparation for Fine-Tuning

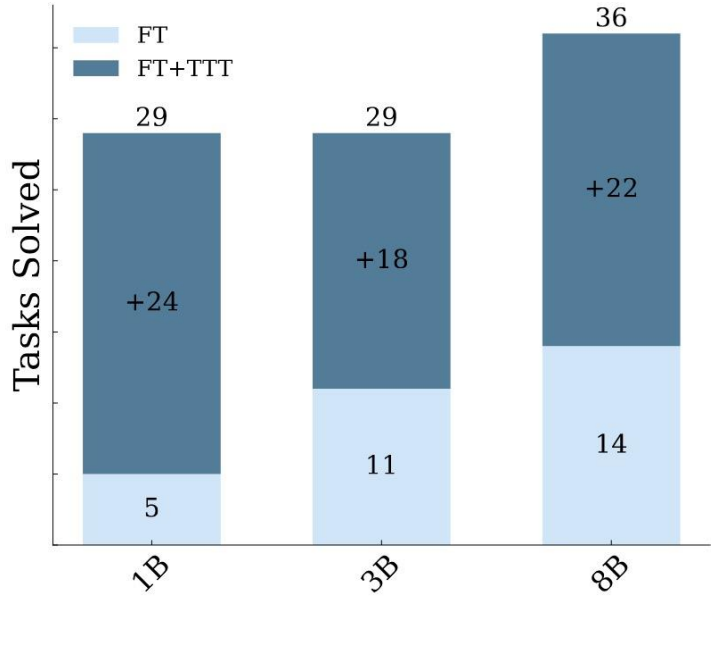
1. **Using Existing Generators:** REARC Generators
2. **Few-Shot Prompting with Large Models:**
   - Use few-shot examples to create new generator functions g'.
   - Uniformly sample m examples from the existing dataset, repeat to produce many tasks.
   - Enhance generators with task descriptions (categories, summaries, and descriptions).
3. **Geometric Transformations**



5.2 Impact of Fine-Tuning Data:

- **Best Performing Data:** REARC and rule-based augmentation yielded the best results.
- **LM-Generated Tasks:** Caused a 5% drop in performance, indicating a need for better filtering mechanisms.
- **Fine-Tuning vs. TTT Performance:** No significant correlation between the two.
- **Fine-Tuning Performance:** TTT enabled smaller models (1B, 3B) to achieve accuracy comparable to larger models.





6. Limitations and Future Directions

Limitations:

- High computational costs
- Limited statistical analysis due to computational constraints.

Future work:

- Optimizing augmentation strategies.
- Exploring broader applications of TTT.

Conclusion

- TTT is a powerful mechanism for improving reasoning tasks.
- Combines test-time compute with task-specific adaptation.
- Bridges the gap between neural and symbolic reasoning approaches.