

Test-time Adaptation of Tiny Recursive Models

Ronan McGovern
Trelis LTD
Trelis.com

1 Introduction

1.1 compute requirements

The ARC Prize 2025 competition allows for the use of four L4 accelerators for twelve hours. As a reference point, TRM pre-training on 1,120 tasks for 100k epochs takes roughly 48 hours on 4xH100 SXM GPUs. Accounting for a factor of roughly 8x between bf16 flops on a H100 SXM versus an L4 accelerator, there is only about $1/8 \times 1/4 = 1/32$ th of the compute necessary to complete full pre-training on this model size. Cast in different terms, such pre-training takes approximately 750k optimizer steps, with a global batch size of 768, while the competition runtime allows for only about 15k optimizer steps at a batch size of 392, given that one must also allow time for inference/evaluation.

The premise of Test-time adaptation is that, by conducting pre-training prior to competition submissions, one can achieve better performance with fine-tuning than one could conducting pre-training from scratch.

2 Methods

A recursive transformer model was pre-trained on ARC AGI II training tasks in close accordance to the Tiny Recursive Model paper [?]. During competition submissions, this pre-trained model was fully fine-tuned on the train example pairs of the test tasks. This fine-tuned model was used to predict test example outputs, using a majority voting method.

2.1 Pre-training

Three models were pre-trained. A first model model was trained almost exactly in line with the original

TRM paper. A second model was pretrained with an expanded pre-training dataset and for double the original number of epochs. A third model was then pre-trained with a smaller dataset, filtered for tasks matching ARC AGI II public evaluation split difficulty.

2.1.1 Original Paper Replication - 100k epochs

A TRM was trained in close alignment with the original paper, but with only 4 lower reasoning cycles instead of 6 reported in the paper. This deviation was unintentional and resulted from a commit in the TRM Github repository using that same value. Interestingly, other ablations by Xin Gao (reference here) and Konstantin Schuerholz also use this value of 4.

The data mix matched that of the original paper and included:

- ARC AGI II Training split [1000 tasks]: train + test example pairs
- Concept ARC split [160 tasks]: train + test example pairs
- ARC AGI II Evaluation split [120 tasks]: train example pairs only (test used for evaluation)

2.1.2 Extended Data for 200k Epochs

Following the heuristic that neural nets often improve in performance with longer pre-training, a second model was pre-trained for double the original number of epochs.

Following the heuristic of more higher in-distribution data helping the performance of neural nets, the dataset was expanded in two ways:

1. Inclusion of evaluation split test example pairs:
The test example pairs from 100 of the 120 public ARC AGI II evaluation split were included

in pre-training. Of the remaining 20 tasks, ten were used as an evaluation split, and ten were used as a post-training test split. Concretely, train example pairs from 110 tasks were included in pre-training and test example pairs from 100 tasks were included in pre-training. Test example pairs from 10 tasks were withheld for evaluation during pre-training, while train AND test example pairs from 10 tasks were withheld entirely for use in post-training.

2. Inclusion of 40 tasks from Simon Strandgaard’s “tama” dataset [?]. These human-reviewed tasks cover concepts typical in ARC challenges and are of a difficulty somewhere between ARC AGI I and ARC AGI II. The hope for including this data was to broaden and enhance the pre-training dataset.

In sum, this meant pre-training on:

- ARC AGI II Training split [1000 tasks]: train + test example pairs
- Concept ARC split [160 tasks]: train + test example pairs
- ARC AGI II Evaluation split [110 tasks]: train example pairs from all 110 tasks and test example pairs from 100 tasks (the other ten serve as evaluation during pre-training)
- tama [40 tasks]: train + test example pairs

While there is the potential advantage of a higher quality, more in-distribution and larger pre-training dataset, there is a large trade-off in being able to assess performance with evaluation and test hold-out sets of only 10 tasks each, particularly when we are trying to assess performance with a granularity of one percent and where the anticipated score is in the region of 1-10 percent.

2.1.3 Filtered Hard Data for 500k Epochs

This third pre-trained variant aimed to test the heuristic that it can be better to train neural nets on a smaller amount of higher quality data for more epochs than on more mixed-quality data for fewer epochs. The same model and hyperparameters were used but training on 110 tasks from the ARC AGI

II evaluation split and 120 hard tasks from the ARC AGI II training split. Training tasks were determined to be hard based on the ability of GPT-5-mini to write a python program that successfully solved all train and test example pairs. Specifically, any task for which GPT-5-mini could write a correct program, given approximately eight attempts, was filtered out. This left 137 remaining ARC AGI II training tasks, from which 120 were selected as a training-hard split. This filtering is somewhat arbitrary and skewed both because it filters based on a) python programming performance and b) LLM, not human, performance. The method was used because of prior unreported work attempting to solve ARC tasks by writing python programs. The choice of GPT-5-mini as a model was made because it was capable of solving only a few ARC AGI II evaluation tasks by writing python programs. As such, if a task can be solved through python program writing by GPT-5-mini this correlates with the task being too easy for ARC AGI II level tasks. And, the motivation for this “hard” dataset split was to have tasks more representative of the semi-private evaluation set on which performance is ultimately graded for the 2025 competition.

In sum, this meant pre-training on 230 tasks:

- ARC AGI II Training split, filtered with GPT-5-mini program writing for “hard” tasks [120 tasks]: train + test example pairs
- ARC AGI II Evaluation split [110 tasks]: train example pairs from all 110 tasks and test example pairs from 100 tasks (the other ten serve as evaluation during pre-training)

2.2 Post-training

Post-training is done at competition time OR in a manner to simulate competition time training using a held out test set.

Four approaches to post-training were conducted:

1. Post-training from scratch. A null hypothesis that is no different than training from scratch, albeit with the limited level of competition-time compute available.
2. Full-fine tuning of a pretrained model.
3. Fine-tuning of embeddings only.
4. Full-fine tuning of a pretrained model, training only embeddings for the first quarter of optimizer steps.
5. LoRA fine-tuning of a pretrained model.

2.2.1 Post-training from scratch.

2.2.2 Fine-tuning of embeddings only.

2.2.3 Fine-tuning of embeddings and then full finetuning

2.2.4 LoRA fine-tuning

3 Results

3.1 Pre-Training

3.2 Post-Training

4 Discussion and Further Ablations

4.1

4.2 On the use of augmentation-specific task embeddings

Both HRM and TRM assign a unique embedding not just to each task, but to every augmented variant (flips, rotations, re-colours) of each task. From the model's perspective, each task variant is an entirely separate task. The model may or may not learn that these tasks are related.

It is interesting to ask whether: a) the model does learn that variants of the same task are related. b) it would be more efficient to use the same task_{id}/embedding for all variants of a given task OR training.

As a means of partially addressing a), one might look at the cosine of the angle between task embeddings for i) variants of the same task and ii) between the base/original/unvaried form of different tasks. If the TRM does indeed learn to encoded task variants similarly, perhaps one should expect a rise in the value cosine between embeddings during pre- and post-training. These cosine measures were charted during training of the second and third pre-trained models, and it was found that: 1. While the cosine does rise away from one, it remains low, suggesting that the model may not meaningfully recognise that variants are related. 2. The cosine of the angle between unvaried base tasks also rises but remains low. This is suggestive of there being little

4.3 On the benefits and need for pre-training

If post training requires fewer optimizer steps than training from scratch - to reach the same performance - then clearly there is meaningful inter-task learning OR at least there are shared concepts involved in training TRM.

The effect can also be understood by pretraining on a single task versus 8 versus 120 tasks on a model of the same size. While it appears possible to achieve similar performance when training on a task individually, or combined with other tasks, it is more efficient - again for a model of fixed size - to jointly train on multiple tasks. Said differently, one can reach the same performance with less compute if one jointly trains on multiple tasks. As such, there appears to be joint concepts to be learned. It is possible of course that much of this joint learning involves primitive concepts relating to grid sizes and colours, as opposed to transformations themselves.

4.4 Tiny Recursive Models as a Variant of Searching Latent Program Space (SLPS)

...

4.5 On the distribution of ARC AGI II tasks

A major challenge in ARC AGI II is that there is little public data that is truly in the distribution of the eventual ARC AGI II semi-private dataset. It is known that any approach scores remarkably similarly on the ARC AGI II public evaluation set and on the ARC AGI II semi-private (and likely private?) dataset. This means that those datasets are closely in-distribution. By contrast, the ARC AGI II training dataset (and the hard split) appears not to be in distribution as scores do not correlate closely with ARC AGI II evaluation sets.

For the purpose of research, it would have been highly useful to have an ARC AGI II training split of 120 tasks in the same distribution as the public eval and semi-private eval set. This would have allowed for pre-training on such a set, followed by post-training on the public eval set to accurately assess performance.

The closest approximation of this would perhaps have been to pre-train on 60 of the 120 public eval tasks, and post-train on the other 60. However, this has two drawbacks: - Statistical power, already small at just 120 tasks, is even smaller when one takes a subsplit - Model performance is sensitive to the amount of data that must be encoded. For the same model size, one cannot directly compare the performance pre or post training on 120 tasks versus 60 tasks.

5 Future Work

6 Conclusion

7 Acknowledgements

This work was supported by Runpod, which provided \$15k of compute, and by Lambda Labs, which provided \$1k of compute. Thanks to Lewis Hemens for collaboration on ARC Prize research and Jack Boylan for assistance in running the pre-training replication.

Table 1: Raw dataset splits used across experiments.

Challenges file	Puzzles	Avg. train inputs	Avg. test inputs
arc-agi-concept_challenges.json	160	2.67	3.00
arc-agi-training2_challenges.json	1000	3.23	1.08
arc-agi-evaluation2_challenges.json	120	2.98	1.43
arc-agi-tama_challenges.json	50	3.18	1.52
arc-agi-test_challenges.json	240	3.20	1.08

Table 2: Derived splits constructed for extended pre-training and evaluation.

Challenges file	Puzzles	Avg. train inputs	Avg. test inputs
arc-agi-evaluation2train_challenges.json	100	2.96	1.44
arc-agi-evaluation2eval_challenges.json	10	2.90	1.60
arc-agi-evaluation2test_challenges.json	10	3.30	1.20
arc-agi-traininghard_challenges.json	120	2.98	1.09
arc-agi-evaluation2clean_challenges.json	114	2.97	1.46

A ARC Task Example Data Splits

Notes on derived splits. The evaluation2train, evaluation2eval, and evaluation2test files are all sampled from the ARC AGI II evaluation split. These subsets supply the pre-training and post-training tasks for the second model configuration.

Side notes.

- Six tasks in the ARC AGI II evaluation split also appear in ARC AGI I. When adapting models pre-trained on ARC AGI I, the arc-agi-evaluation2clean_challenges.json split filters these duplicates to avoid contamination.
- The placeholder arc-agi-test_challenges.json split contains fewer test examples than the evaluation set. Inference on this set is roughly 33% faster than the final competition rerun and can under-estimate runtime, risking notebook timeouts during submission.