MindsAI, Tufa Labs
ARC Prize 2025 Solution

Written by: Jack Cole
Mindware Consulting, Inc.
jackcole@mindware.mobi
X: @mindsai_jack

Team: Jack Cole, Dries Smit, Isaiah Pressman, Mohamed Osman, Michael Hodel

Team Bios:

Jack Cole is an AI Researcher, Clinical Psychologist, and App Developer. He has had a dual career in Clinical Psychology and various technology-related businesses. He holds a PhD in Clinical Psychology and works in a part-time private practice. His top app, a brain training app called Mind Games, has had over 30 million downloads. His psychology research background informs his approach to AI Research. After leading his ARC team to first place in the 2023 ARCathon, his team held SoTA scores for ARC-AGI-1 in 2024. The 2024 team discussed their approach with Tim Scarfe on Machine Learning Street Talk. He has driven pioneering approaches to ARC including TTT and AIRV, which has inspired innovation in test-time adaptation. His team was the first to successfully use machine learning for ARC to beat the record held by symbolic approaches for 5 years. He worked under contract at Tufa Labs as an AI Researcher for the last year and is now working independently under his company, Mindware Consulting, Inc. He developed this year's solution and has worked full time on ARC for the past 3 ½ years (minus approximately 6 months at the end of 2024 through mid 2025). He has conducted hundreds of experiments on ARC and conducted extremely long training runs (up to 2 ½ years).

Dries Smit is a Machine Learning Researcher specialising in reinforcement learning, multi-agent systems, and large language models. He holds a PhD in Electrical and Electronic Engineering and has applied advanced AI techniques across both medical and financial domains. His work spans large-scale model training, distributed compute, and adaptive reasoning systems. Dries led the team that won first place in the ARC-AGI-3 Preview competition, adapting curiosity-based approaches to achieve state-of-the-art performance. He has a strong track record of designing multi-agent frameworks and fine-tuning large models for practical applications. Notably, he led the development of Laila, a fine-tuned variant of Llama 3.1 designed to assist biologists by interfacing directly with laboratory equipment. He currently conducts research on large-scale reinforcement learning and test-time adaptation, contributing to Tufa Labs' efforts to bridge the gap between foundation models and general reasoning systems.

Isaiah Pressman is an AI researcher specializing in multi-agent reinforcement learning and large language models. His previous work includes research in applying computer vision to histopathology slide analysis, building the ML and data pipelines as the founding engineer for a dynamic pricing startup, and four top-2 finishes in Kaggle competitions. He currently works at Tufa Labs, researching improving large language model training techniques and reasoning capabilities.

Mohamed Osman is an ML practitioner and researcher based in Calgary, Alberta, Canada, holding a Master's degree in Electrical Engineering. With over five years of experience in machine learning setups and research projects. He was drawn to the ARC Prize because of his interest in the problem statement and agreement with the competition's definition of intelligence. Although he joined the team with the intention of collaborating, outside factors prevented this opportunity from fully materializing, and he ultimately spent very little to no time on the solution. He teamed up with the other members based on prior collaborations in similar competitions. He was a co-developer of TTT for ARC working with Jack Cole in 2022/2023.

Michael Hodel is an AI researcher well known as the creator of RE-ARC, a dataset for procedural example generation, and ARC-DSL. He has experience in machine learning and prior work on ARC, including contributing synthetic datasets to the team's training corpus in 2024. His inclusion on the 2025 team reflects his long-standing involvement and previous collaborations with the members. While he has had little involvement in this year's solution and spent limited time on the project, his foundational contributions to the ARC community remain significant.

Introduction:

For the 2025 season the following components were used: TTT, AIRV, Ensembling, new augmentations, tokenizer BPE dropout, and fully-sharded data parallelism. The final score for the 2025 season was 15.42% with a rank of 3rd place on the leaderboard. In addition to the components mentioned above, many additional methods were tried that yielded beneficial, but not additive effects. This means that these techniques help as alternatives to TTT/AIRV, but not additively when employed in concert. Three examples of things tried that helped, but not additively: refinement (tried several methods), TTRL (used DPO on beam search pairs to try to boost the correct answers), ARC 2 targeted training data, model merging, and Monte Carlo dropout during inference.

Competition Methods:

- Large Scale/Long Timespan Training (TPU for up to 2 ½ years)
- Diverse Training Objectives and Dataset Extending Beyond ARC
- Dataset Contains 100M+ Training Examples (70M ARC tasks)
- Dataset Extended for ARC-AGI-2 (Augmented Training Dataset and 27 ARC 2 Style Synthetic Tasks)
- T5 Span Corruption Objective (During Training)
- Reversal Augmentation (During Training)
- Test-Time Training (TTT) on the Entire Test Set

- Augment, Inference, Reverse Augmentation, Vote (AIRV)
- Ensembling (Two checkpoints of a 660M Salesforce Code T5 model)
- Tokenizer BPE Dropout
- Span Corruption Refinement during TTT (with Second Model)
- Fully Sharded Data Parallelism (FSDP) using HuggingFace Accelerate

Methods for Ablation Runs:

In addition to describing how the solution works, ablations were performed to demonstrate the effects of the various components of the solution. For the ablation studies, this author used a strong small model based on the Salesforce Code T5 family (77M parameter) that was trained on our large dataset (ARC-AGI Mega) for approximately 2 years on v2-8 and v3-8 TPUs. It also trained for approximately 7 days on a v4-64 TPU. We release this model as a useful research artifact. The 77M model was used for the ability to conduct more rapid testing on consumer grade GPUs, which was used after this author's contract with Tufa Labs ended in October, 2025. This author chose to use a test dataset we call ARC 1.5, which was created in 2024 by Michael Hodel. The reasons for its use are: ARC-AGI-2 yields a lower signal granularity due to the difficulty level and ARC-AGI-1 eval items are included in training minus the test items (also including RE-ARC Eval, which makes the ARC 1 evaluation set IID). Unless otherwise noted below, runs used 5 repetitions of each condition and averaged the scores across those runs. This was done to produce more robust comparisons between the conditions and for the calculation of confidence intervals.

Test-Time Training (TTT or TTFT)

Cole (2022) first proposed test-time fine tuning (TTFT) for ARC based on training on retrieved tasks with high cosine similarity. In 2023, we made critical discoveries that made TTFT work for ARC winning the ARCathon 2023 competition (Cole & Osman, 2025). All top solutions in 2024 used the techniques we shared in blogs and interviews (with additional innovations by the contestants). To this author's knowledge, the notion of Test-Time Training (TTT or TTFT as we have sometimes called it) was introduced in the domain of computer vision by Sun et. al. (2020). For ARC, this author's crucial discovery was the use of each example pair as a test pair to create new ARC items (though reduced by one demonstration example). We have called this the task permutation method. The permutation method enables TTT by providing labels for supervised fine tuning. After permuting the tasks with this method, we further augment them using geometric augmentations (rotations and flips) and color permutations. New for this year is the use of mixup and combination augmentations (see section Augmentation Ablations for more details). During the ablation runs, TTT gained 430.0% over 0-shot without beam search and 110.0% compared with beam search with 4 generations on the ARC 1.5 test set. In the competition, we use around 45K training examples for TTT and run training for all items at once. We use full fine tuning rather than PEFT techniques such as LoRA. Recently, this author discovered LoRA can work well with our models, but is slightly less performant. Another strong technique is layer-wise fine tuning (updating parameters in only certain layers in the model).

Again, layerwise training can produce similar performance, but requires a lot of tuning to optimize (e.g., which patterns of layers to include for fine tuning).

Augment, Inference, Reverse-augmentation, Vote (AIRV)

Another strong technique we introduced in 2023 was AIRV. It can be considered a form of Test-Time Augmentation (TTA), first proposed by our 2023 teammate, Matteo Batelic. Used in computer vision, the approach often involves augmentation, inference, reverse augmentation, and merger (e.g., through pixel-wise averaging; e.g., Moshkov, et. al., 2020). When used with classification problems, no reversal is used. For ARC, the key innovation is storing the original augmentation and reversing that augmentation after performing inference. Finally, board-level voting seems to be the best based on prior unpublished work (though it may be there are better methods, but we have not found them). Other methods were tried such as row-wise or pixel-wise voting and logit averaging (without positive results above board-level voting). On the ARC 1.5 test set, AIRV gained 412.5% over 0-shot without beam search and 103.0% compared with beam search. In the competition we use 10K augmented tasks for the inference items.

Ensembling/Self-Ensembling

Ensembling refers to the machine learning technique of combining the predictions from multiple models to produce a more accurate and robust final prediction than any single model could achieve alone. During the 2024 season, we found that combining predictions across various models (even the same model multiple times) significantly mitigated individual model failures and improved overall generalization performance. This self-ensembling approach, specifically, involves running the same model multiple times with different stochastic elements (like augmentations, dropout, tokenizer dropout) and aggregating the results. During the 2025 season our best results came from combining the top performing checkpoint of the model with a more recent checkpoint. We have frequently observed during the course of training that a model will get different items correct at different points in training, despite showing an overall increase in the number of correct items. For ensembling, It seems to be helpful if models have somewhat similar performance. Models of different strength can be combined by weighting or implicitly weighting the inference items by using a smaller number of inference samples. For the ablation runs, the solution gains 307.9% with 3 self-ensemble runs vs. 261.4% with no ensembling.

One may speculate that just drawing more inference examples and additional TTT examples could be equivalent (or better) than self-ensembling. To examine this, a more optimal number of TTT/AIRV items was determined empirically by running different amounts. One can see from Figure 3 that using 90K/30K items was equivalent to using 180K/60K items. Using 60K/20K in 3 self-ensemble runs was maybe slightly better, though not significantly so. However, using 90K/30K with 2 self-ensemble runs, produced significantly higher scores than all of the other runs, despite using an equivalent number of TTT/AIRV items. This is a compute matched gain of 6.2%. One interpretation for this is that the different random seeds allow the model to explore a different area of the distribution since the augmented training and inference data would have differences across the self-ensemble runs.

Figure 1. Self-Ensembling Gains

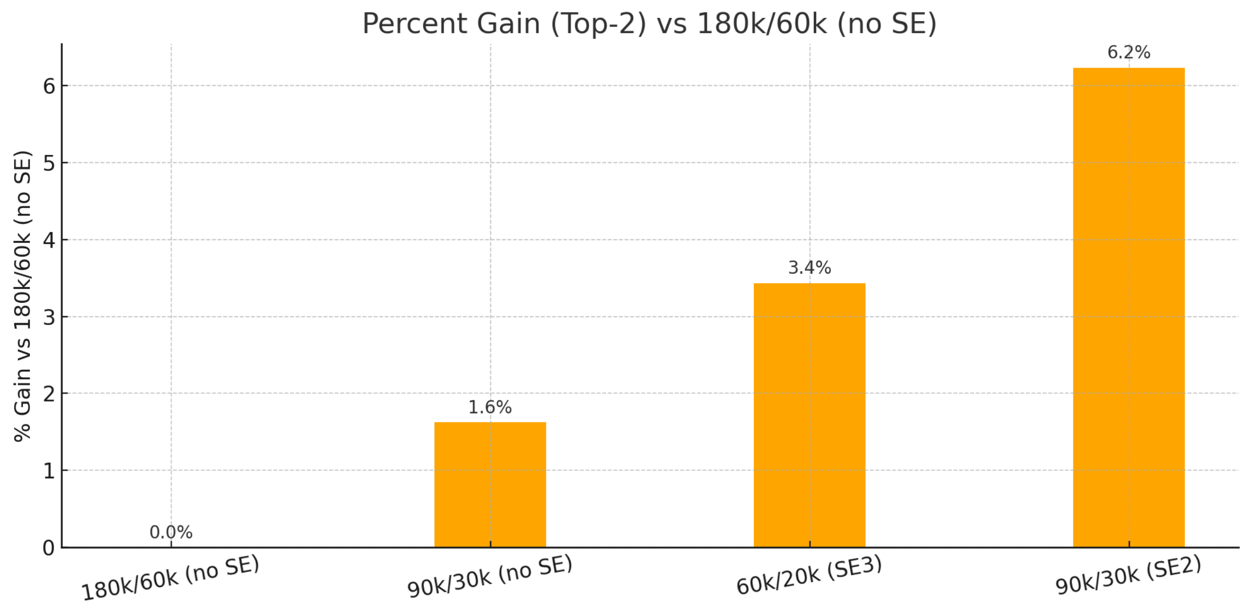
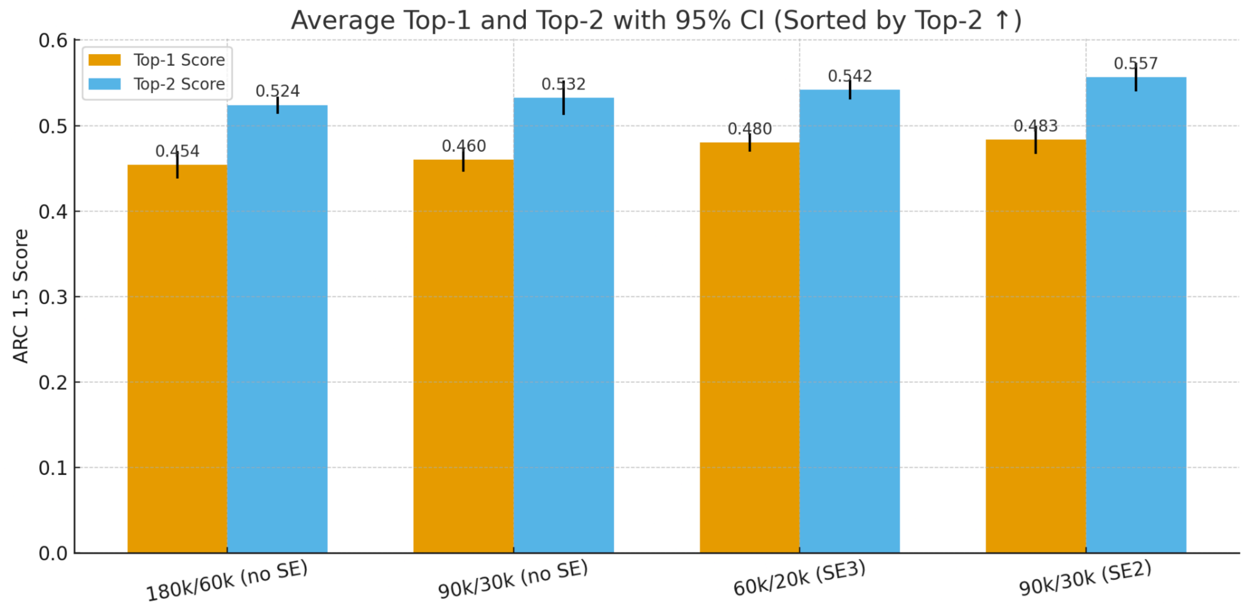


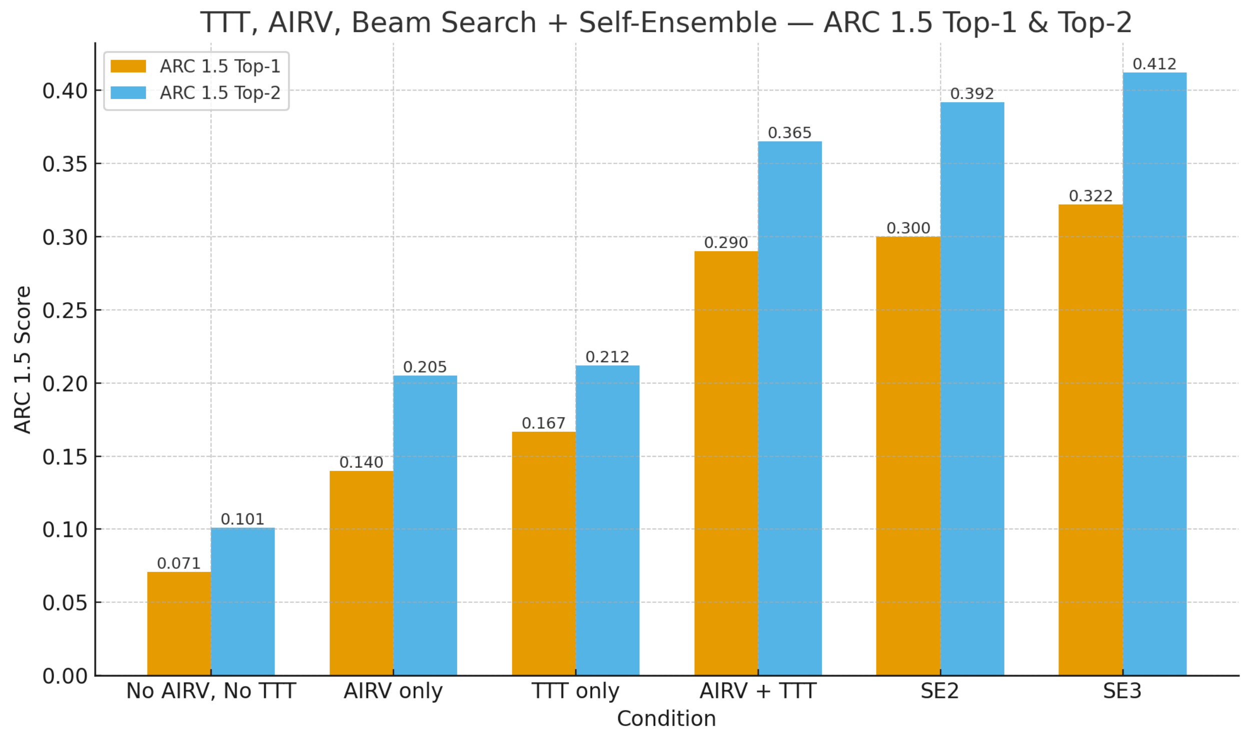
Figure 2. Self-Ensembling Scores



<u>Combined Results</u>

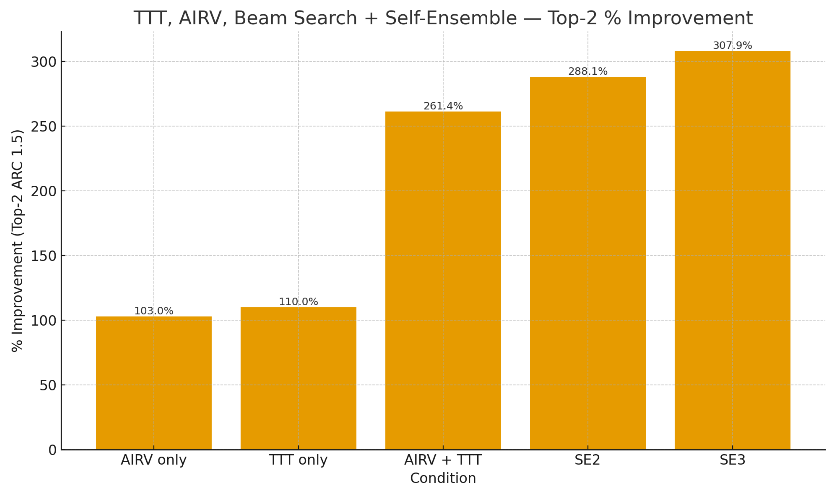
Combining TTT and AIRV shows an additive benefit (almost precisely). The combined gain was 261.4% versus 103.0% for AIRV alone and 110.0% for TTT alone (Figure 4). When compared to 0-shot (no beam search) performance, the combined gain was 812.5% versus 412.5% for AIRV and 430.0% for TTT. Finally, gains after the third self-ensemble run was 930%. Note that using more TTT training examples and inference examples gives more gains than self-ensembling if the number of items chosen is not optimal. Relative gains from more performant 0-shot models tends to be less, but still very strong. It should be noted that the gains reported here for TTT and AIRV are much lower than optimal as low values were used for the number of TTT items and AIRV items. A more optimal run using 90K TTT and 30K AIRV items yields a score of 54% and a gain of 1250% over 0-shot. For reference, our first model in our solution scores 75% on ARC 1.5 with 40K TTT and 10K AIRV items (CodeT5 660M; 2 self-ensemble runs).

Figure 3. TTT, AIRV, Self-Ensemble Ablations



Note: The “No AIRV, No TTT” condition uses beam search with 4 generations and 4 beams. Pure 0-shot scores 4%.

Figure 4. TTT, AIRV, Self-Ensemble Improvement Over Baseline (Beam Search Only)

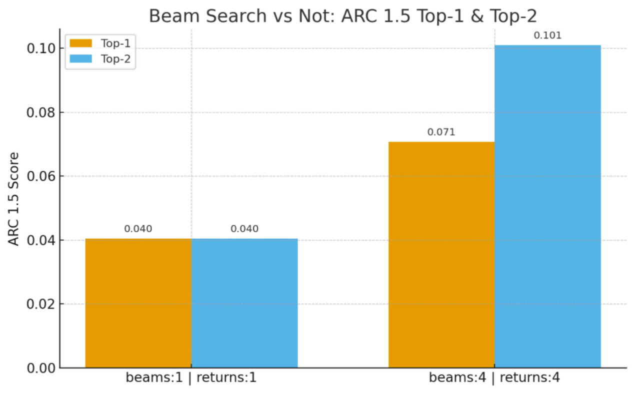


Beam Search

Unlike greedy decoding—which selects only the single most probable next token at each step—beam search maintains a set of k candidate partial sequences (beams). At every decoding step, each beam is expanded, and the top-k sequences (by cumulative probability) are retained. This allows the model to consider multiple promising continuations simultaneously, increasing the likelihood of finding a globally better solution than greedy decoding.

Applied to ARC 1.5, beam search produced a 175% improvement over our 0-shot baseline when using a beam width of 4 and returning 4 candidate generations (see Figure 5).

Figure 5. Beam Search vs. 0-Shot Performance

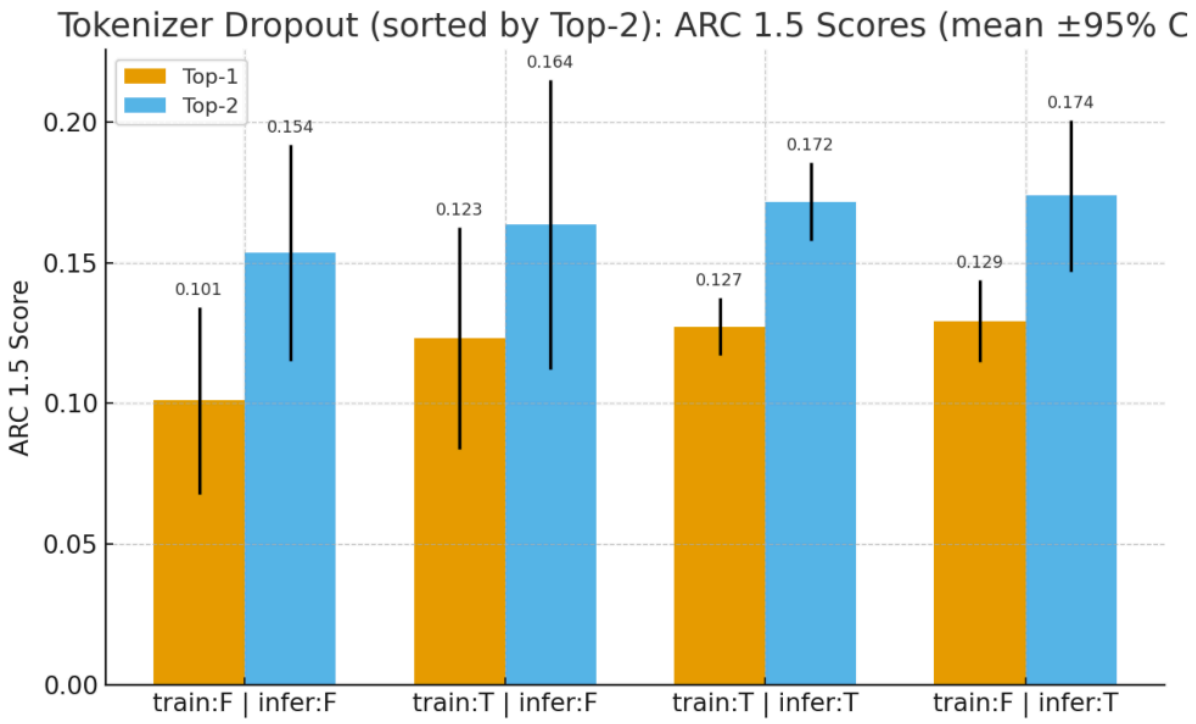


Note: No augmentation and no TTT (one run only b/c deterministic)

BPE Tokenizer Dropout:

This year's solution applies BPE dropout during both test-time training (TTT) and inference. In our ablation, we evaluate stochastic subword tokenization, where merge operations are skipped with probability p (0.2 for TTT and 0.05 for AIRV in this example). Using dropout during training, inference, or both improves ARC 1.5 top-1 and top-2 scores relative to deterministic tokenization, indicating increased robustness to formatting variation. Although prior work typically applies BPE dropout only during training, our results show that enabling it at inference can also be beneficial. During TTT, label tokens are excluded from dropout to avoid degrading supervision. Overall, these findings suggest that subword sampling is an effective, architecture-free regularizer that improves generalization.

Figure 6. Tokenizer Dropout



Augmentation Ablations:

Our core augmentation set consists of **geometric transformations** (rotations and flips) and **color permutation**. Previous attempts to design additional augmentations offered little measurable benefit. This year, we introduce four new augmentations:

- **Mixup** – blends boards from multiple tasks into a single composite example.
- **Combine** – duplicates the input board and applies independent color modifications.

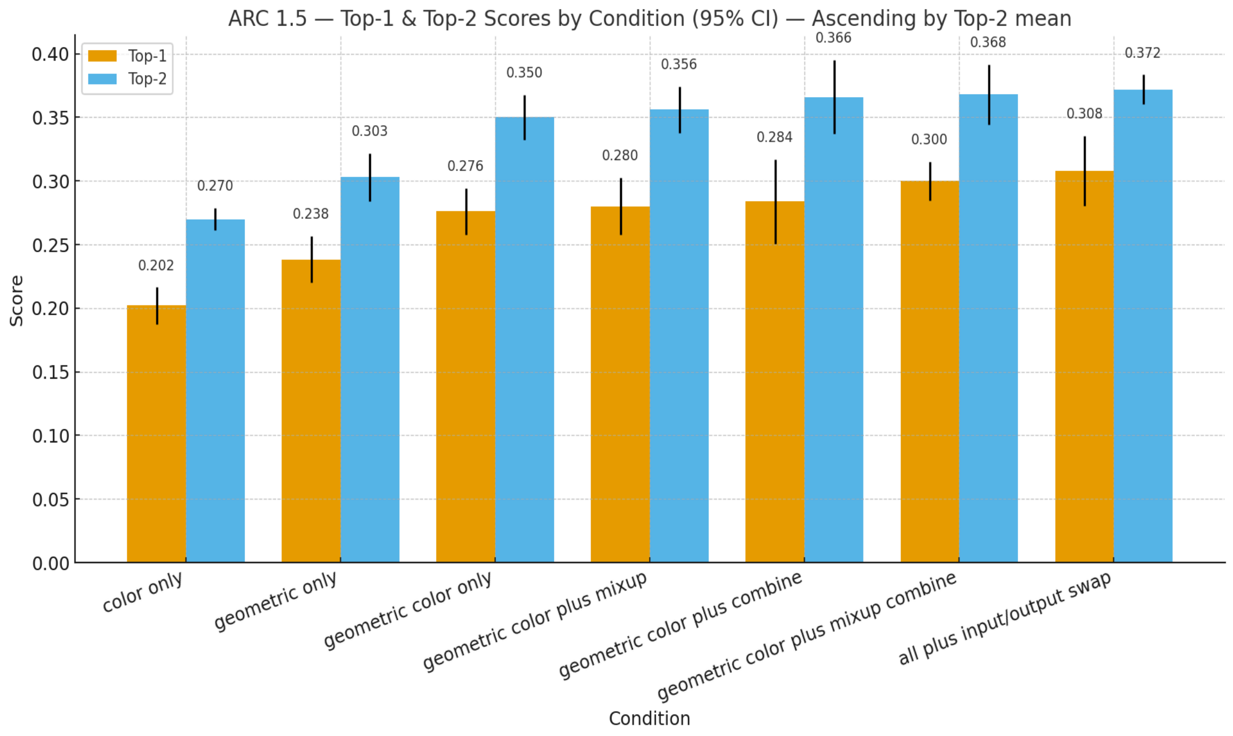
- **Combine-mixup** – mixes boards from multiple tasks *without* merging them (non-overlapping combination).

- **Input/Output swap** – swaps input and output boards in 30% of training samples.

Geometric and color-permutation augmentations remain the strongest contributors to performance. However, the new augmentations increase ARC 1.5 **Top-2 accuracy by 6.3%** in ablation studies (Figure 7). We apply the same augmentations for both TTT and AIRV, except for mixup, which cannot be reversed during inference.

For reversible augmentations (e.g., combine), predictions are separated into their original components before reversing the transformations. Because transformers operate over flattened 1-D token sequences, geometric augmentations may serve as being analogous to viewing the task from multiple spatial perspectives, reducing orientation-specific biases. Color permutation mitigates tokenization artifacts and reduces reliance on specific color–token associations. Mixup appears to help the model form abstractions by exposing it to concept combinations that never co-occur naturally within a single ARC task.

Figure 7. Augmentation Ablations



Model:

We began with the **Salesforce CodeT5-Large** model. Deep into training (approximately 1 year), we progressively reduced the decoder depth from **24 layers to 16**, removing 1–2 layers at a time and allowing the model to recover between reductions. This was performed over the course of 1 month. The encoder remained at the full 24 layers. Across several experiments, we found that removing encoder layers significantly degraded performance, while removing decoder layers had comparatively minor impact. This result aligns with prior findings from Google (Xue et al., 2022), who observed that encoder depth contributes disproportionately to model quality, and that decoder sizes approximately one-third the size of the encoder can be optimal. Google's more recent **T5Gemma** (Zhang et al., 2025) independently replicates this conclusion.

We have consistently found **encoder–decoder architectures** to perform better on ARC-type reasoning tasks, though some decoder-only models can perform well as other contestants demonstrated in 2024. After pruning, our model contained **660M parameters** (down from the original 770M). We ultimately used **two checkpoints** of this same model:

1. an earlier checkpoint (strong on ARC 2 submission), and

2. a later checkpoint trained on data the same data plus additional data targeted toward ARC-2 (augmented ARC-2 eval and synthetic ARC-style items).

During continued training of the second checkpoint, we applied **three refinement methods**, inspired by work done by Dries Smit. Dries Smit demonstrated a refinement-centric model trained from a strong base of the 660M model above achieved relatively strong zero-shot performance but benefited less from TTT/AIRV; Isaiah Pressman explored a diffusion-based approach that appeared promising, but we were not able to fully evaluate it before the contest ended.

Model Training:

Training was performed on **Google TPU Research Cloud** hardware, whose TPU resources we gratefully acknowledge. Our training data expanded beyond the data we described in our paper. Additions included:

- **ARC-AGI-2 Training + Evaluation (augmented)**
  (The ARC-AGI-2 evaluation portion was not used for the first released model `mindware/arc-codet5-660m` nor the small model, but it ***was*** *used for `mindware/arc-codet5-660m-scr`). It does not appear that this additional data helped the score.*

- Reasoning Gym (Stojanovski et al., 2025)
  – ~300k examples across multiple reasoning domains (included in the archive).

- **ARC-style synthetic data (27 task types)**
  – Inspired by structures in ARC-AGI-2; all included in the archive.

A key improvement this year was the introduction of **augmentation-driven data expansion**. One model that had stagnated for ~4 months of continued training achieved a ~**100% gain in zero-shot performance** (ARC-AGI-1 public test and ARC-1.5 test) after these augmentations were introduced.

The new augmentations allow the model to infer task directionality using **prefix cues**:

Augmentation type	Description	Applied to
Prompt reversal	Reverse prompt sequence	5% of samples
Answer reversal	Reverse answer sequence	5% of samples
Both reversal	Reverse both prompt and answer	5% of samples
Character reversal	Reverse answer at character level	10% of samples

Reversals were performed in two modes:

- **Character-wise reversal** – reverses every character in the output string (applied to answers only).

- **Word-wise reversal** – reverses tokens separated by spaces.
  For ARC, this means the model sees output boards with reversed row order and board metadata positioned last.

All augmentations produced **new training examples** rather than modifying originals in place.

In prior years, we used a **custom span-masking objective**: removing 1–15% of characters and replacing them with `<answer>`, prompting the model with *"What is `<answer>`?"* The correct answer was included in ~40% of prompts. This year, the original **T5 span corruption objective** (Raffel et al., 2020) was applied to 15% of examples, with a mean span length of 3. Both methods were independently beneficial. The original span corruption has the benefit of aligning with pretraining and corrupting multiple segments.

Additional dataset contributions not listed in Appendix A that were added in 2024 (Cole & Osman, 2025):

- In 2024, **Michael Hodel** generated synthetic datasets based on ARC-AGI-1 eval and another synthetic data corpus he called ARC 1.5.

- **Andreas Köpf** contributed post-competition (2024) synthetic concepts generated via foundation models using Hodel's (2024) RE-ARC data as seed (estimated 2,000–3,000 new concepts with verifier-checked outputs).

Dataset Scale and Release

In total, the training corpus exceeded **100 million examples**, of which approximately **70 million are ARC tasks**. We release this dataset publicly on Hugging Face as **`ARC-AGI Mega`** for the purposes of reproducibility and as a research artifact. Datasets from other Huggingface authors are filtered out from the upload but referenced in our model cards.

Training Pipeline

We include the **Flax training script** and **Hydra configuration** used to train CodeT5-Large under `tpu/` in the repository. Additional datasets are included there as well (not part of ARC-AGI Mega) and were mixed according to config-defined sampling ratios.

The training script loads datasets from disk; users wishing to train on the released dataset must modify the script to load directly from Hugging Face. They may also download the raw csv.gz files to use without code modification.

Conclusion

The 2025 MindsAI/Tufa Labs solution is best understood as a reasoning system built around test-time adaptation rather than a static model. The approach mitigates some of the weaknesses of transformer models, but ARC-AGI-2 proves to be at least somewhat adversarial to the approach highlighting the need for further innovation. TTT, AIRV, and self-ensembling work together to let the model actively explore the local task distribution at inference time, rather than relying solely on what was learned during pretraining and finetuning. This author believes that dynamic systems like our approach are more biologically plausible as most biological neural networks are constantly updating. Our ablations on ARC 1.5 show that each component provides large, quantifiable gains, and that their combination produces nearly additive improvements in accuracy—especially when TTT/AIRV budgets and ensemble sizes are matched on compute.

On the training side, the results highlight three elements that matter for ARC-style reasoning: (1) large, diverse, ARC-focused synthetic and augmented corpora (ARC-AGI Mega), (2) encoder–decoder architectures that preserve a deep encoder while trimming the decoder, and (3) augmentation-driven data expansion, including reversals and BPE dropout, to force the model to represent structure rather than surface form. Together, these choices produced models that can be pushed much further at test time via TTT and AIRV than their raw 0-shot scores would suggest.

This author hopes that the release of the solution code, training scripts, and training data will spur further innovations as we step towards AGI.

Acknowledgements

Google TPU Research Cloud contributed TPU resources to this project. It would not have been possible to develop these groundbreaking approaches without their generosity.

Lambda Labs contributed $12500 in compute credits towards our work. I am very appreciative of their contribution.

References

Cole, J. [Mindware Consulting]. (2022, November 2). ARTI - An ARC solver architecture [Video]. YouTube. https://www.youtube.com/watch?v=Am_4c0y62iM

Cole, J., & Osman, M. (2025, June 17). Don't throw the baby out with the bathwater: How and why deep learning for ARC. arXiv. https://doi.org/10.48550/arXiv.2506.14276

Hodel, M. (2024). *Addressing the Abstraction and Reasoning Corpus via procedural example generation*. arXiv. https://doi.org/10.48550/arXiv.2404.07353

Moshkov, N., Mathe, B., Kertesz-Farkas, A., Hollandi, R., & Horvath, P. (2020). Test-time augmentation for deep learning-based cell segmentation on microscopy images. Scientific Reports, 10, 5068. https://doi.org/10.1038/s41598-020-61808-3

Stojanovski, Z., Stanley, O., Sharratt, J., Jones, R., Adefioye, A., Kaddour, J., & Köpf, A. (2025). *Reasoning Gym: Reasoning environments for reinforcement learning with verifiable rewards* (Version 2). arXiv. https://doi.org/10.48550/arXiv.2505.24760

Sun, Y., Wang, X., Liu, Z., Miller, J., Efros, A. A., & Hardt, M. (2020, July 1). Test-time training with self-supervision for generalization under distribution shifts [Preprint]. arXiv. https://doi.org/10.48550/arXiv.1909.13231

Xue, L., Barua, A., Constant, N., Al-Rfou, R., Narang, S., Kale, M., Roberts, A., & Raffel, C. (2022). ByT5: Towards a token-free future with pre-trained byte-to-byte models (Version 3). arXiv. https://doi.org/10.48550/arXiv.2105.13626

Zhang, B., Moiseev, F., Ainslie, J., Suganthan, P., Ma, M., Bhupatiraju, S., Lebron, F., Firat, O., Joulin, A., & Dong, Z. (2025). Encoder-Decoder Gemma: Improving the quality-efficiency trade-off via adaptation (Version 1). arXiv. https://doi.org/10.48550/arXiv.2504.06225