IMPLICIT RULE INDUCTION WITH TEST-TIME TASK EMBEDDINGS IN ARC-LIKE TASKS

Anonymous authors
Paper under double-blind review

ABSTRACT

The Abstraction and Reasoning Corpus and related benchmarks evaluate whether AI models can solve novel reasoning tasks, but often leave unclear whether success reflects inference of the intended underlying rule or reliance on shortcuts. We address this gap by studying test-time task embeddings in Vision ARC (VARC), a model in which a pre-trained backbone is complemented by a trainable embedding representing the transformation rule. In VARC, test-time training (TTT) is jointly applied to the backbone and task embedding. Here we introduce a novel two-step TTT protocol: first finetune only the task embedding (Embed-TTT), then freeze it and finetune the backbone. Across ARC-AGI-1, ConceptARC, and two controlled datasets with known rules, Embed-TTT consistently yields improved task embeddings, ones that align better with underlying task rules, improve embedding-based retrieval, and enable accurate linear probing of known rules. Qualitatively, Embed-TTT identifies more semantically meaningful relations between test and train tasks. We also show that optimizing only task embeddings (less than 0.01% of model parameters) already solves a non-trivial fraction of ARC-AGI-1, ConceptARC and Mini-ARC tasks, while the full two-step pipeline improves final performance. Finally, we show that Embed-TTT recovers the underlying geometric structure of parametric rules and learns compositional capabilities that enable rule interpolation, but not extrapolation. These findings support a clearer separation between rule induction and rule execution in ARC-like evaluations, motivating benchmarks that better distinguish in-distribution from out-of-distribution rules.

1 INTRODUCTION

The Abstraction and Reasoning Corpus (ARC) was introduced to evaluate a form of humanlike fluid intelligence that is difficult to capture with standard machine learning benchmarks (Chollet, 2019). In each task, a model is given only a few input-output grid demonstrations and must infer the transformation rule that maps inputs to outputs. This setup has inspired a growing family of ARC-like benchmarks for studying humanlike abstraction, few-shot adaptation, and systematic generalization. However, standard ARC-like evaluations typically measure whether a model produces the correct output, not whether it has inferred the intended rule. This distinction is important: a model may solve a task by inducing the underlying rule, but it may also exploit dataset-specific shortcuts, memorize related patterns, or rely on test-time optimization without forming a reusable representation of the task (Beger et al., 2025). Conversely, a model may identify a useful rule but fail to execute it perfectly. As a result, task accuracy alone conflates rule induction, execution and memorization. This issue is amplified by the fact that most ARC-like datasets do not provide the rules or generators underlying their tasks. Consequently, it is usually unclear whether rules underlying test tasks are in or out-of-distribution with respect to the rules known from training tasks, where training tasks would be drawn from the ARC-AGI-1 training set and test tasks from ARC-AGI-1 evaluation set, for example. From a classical machine learning perspective, this makes benchmark results of models trained on the training tasks difficult to interpret: strong performance may reflect genuine abstraction, but it may also reflect favorable overlap between train and test rules. We therefore argue that ARC-like datasets and evaluation should more explicitly study whether models induce meaningful task representations, and should distinguish rule induction from rule execution whenever possible.

In this paper, we study implicit rule induction through the task embeddings of Vision ARC (VARC) (Hu et al., 2026), a ViT-based model that represents each task with a trainable embed-

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.

ding, which steers the output grid generation in a separate model backbone via a latent representation of the underlying rule. We show that the default VARC test-time training procedure, which jointly finetunes the task embedding and the model backbone, produces poorly aligned test-train task embeddings that do not retain embedding space information from the training stage. We propose a simple alternative, Embed-TTT: at test time, we freeze the backbone and optimize only the task embedding. This forces test tasks to be represented in the same embedding space as training tasks, as both pass through the same frozen backbone, making the resulting embeddings more suitable for analyzing whether the model has identified the underlying rule. We evaluate these embeddings on ARC-AGI-1 (written ARC-1 for brevity) (Chollet, 2019), ConceptARC (Moskvichev et al., 2023), and two controlled ARC-like datasets with known rules. When ground-truth rules are unavailable, we test whether duplicated training tasks recover their original embeddings. When rules are available, we evaluate whether ground-truth rule labels or parameters can be decoded from embeddings using linear probes. Across these settings, Embed-TTT yields substantially more rule-aligned embeddings than default full-model test-time training, while also supporting increased task-solving performance when followed by optional backbone finetuning. Overall, we make the following **contributions**:

- We propose a novel test-time training protocol, Embed-TTT, for VARC, which produces semantically meaningful task embeddings for ARC-like tasks. To evaluate Embed-TTT against the default full model TTT, we further introduce an evaluation methodology for implicit rule induction.
- We show quantitatively that Embed-TTT embeddings align better with task rules across four ARC-like datasets, and we provide qualitative evidence of meaningful test–train semantic similarity in ARC-1. In addition, optimizing only task embeddings solves 17% of ARC-1 test tasks, and nearest-neighbor concept transfer from ConceptARC to ARC-1 yields 36% correct concept associations, without training for that transfer objective nor for concept classification.
- We analyze the task embedding space on a specific case and characterize which unseen tasks remain solvable. We show that task embeddings recover the dataset geometry and support compositional rule interpolation, reframing ARC-like performance through a standard in-/out-of-distribution lens.

2 Related work

2.1 Problem statement

Definitions. An *ARC-like dataset* is a set of *tasks*, and a task is a set of (x, y) pairs governed by a deterministic *rule* that maps each *input* x to its *output* y. In theory, the rule can range from a simple function to a complex algorithm, but implicitly it is commonly accepted that the rule should be expressible in natural language and understandable by human readers. Following the original ARC-1 format, each x and each y is a two-dimensional grid of integers, such that a task can be viewed as a visual puzzle, where each cell of the grid is colored according to its integer value. Typically, each task is split into demonstration pairs (x_i, y_i) and a test input x', such that a solver should infer a common rule R that maps each x_i to y_i and then should apply R to map the test input x' to y_i. To elicit in-context test-time learning, an ARC-like dataset is usually split into training tasks and test tasks, such that one should learn generic task-solving strategies on the training tasks, and successfully apply them on the test tasks without any human intervention.

Performance. The performance of a model on an ARC-like dataset is the proportion of test tasks solved. A test task is solved when the test input x' is correctly mapped to its ground-truth output y', which means reaching 100% cell-wise accuracy on the y'. In practice, several predictions can be made for a given test input x', and the task is considered solved if one of two predictions matches the ground truth ("pass @ 2"). In general for ARC-like benchmarks, while ground-truth outputs y' are given for each task, the explicit rules underlying the tasks are not provided, which makes it difficult to evaluate whether the model actually learned the intended logic behind the tasks.

2.2 Approaches

Implicit and explicit reasoning. Several works have pointed out the tendency of large language models (LLMs) to rely on shortcuts or shallow reasoning on tasks requiring analogy or abstraction (Beger et al., 2025; Mineault et al., 2026; Lewis & Mitchell, 2025). Many benchmarks targeting these abilities are susceptible to such behavior, as they primarily evaluate the correctness of the final output rather than the actual reasoning process. Recent work has therefore shifted toward learning

more robust representations, either internally through structured latent models such as World Models (Piriyakulkij et al., 2025; Maasch et al., 2025), or externally via explicit reasoning artifacts, for instance in program synthesis approaches (Langenfeld et al., 2026; Li et al., 2025). In this work, we aim to bridge these perspectives by enforcing the model to store its internal representation in an explicit vector that can be directly extracted and "executed," for instance through linear projections.

Embedding vectors. The extraction of embedding or activation vectors is a common technique in interpretability research. Todd et al. (2024) identify sets of attention heads that implement simple functions, such as word translation or antonym mapping, and show that combining such function vectors yields non-trivial compositions of functionality. We extend this line of work to models applied to abstract reasoning tasks by introducing a test-time tuning regime that encourages the formation of explicit task embeddings. We analyze the extent to which these embeddings capture underlying rules, including their compositional properties and limitations. We also study similarities between embeddings of tasks that share related rules. Related observations, albeit in a more correlational setting, have been reported by Lake & Baroni (2018); Hill et al. (2019); Veldkamp et al. (2023).

Test-time adaptation in abstract reasoning tasks. Test-time training (TTT) has been used in several high-performing systems for solving ARC-1 tasks (Li et al., 2025; Chollet et al., 2024; Hu et al., 2026; Akyürek et al., 2025). TTT typically works by generating additional training examples through simple augmentations at test time. However, while TTT improves model performances on ARC-1, it remains unclear whether it leads to the induction of meaningful task representations or instead relies on unconstrained parameter updates. One of the motivations of our work is to study this distinction, by constraining TTT to operate through explicit task embeddings rather than full model finetuning.

Vision ARC (VARC). Many of the strongest models on ARC-like datasets are closed-source LLMs, whose internal representations cannot be inspected. A rare exception is Vision ARC (VARC) (Hu et al., 2026), a ViT-based model trained from scratch that explicitly represents each task through a trainable task token, numerically encoded as a 512-dimensional *task embedding*. We focus on VARC because its code is provided and it is usable with limited academic resources. VARC also has motivated follow-up approaches such as LoopViT (Shu et al., 2026). During training, VARC solves tasks as image-to-image translation problems conditioned on their corresponding task embeddings. At test time, the task embedding is randomly initialized and learned from the test examples, together with finetuning of the trained backbone. We refer to this procedure as the baseline Full-TTT approach. We considered another popular scratch-trained model, the Tiny Recursive Model (TRM) (Jolicoeur-Martineau, 2025), but preliminary experiments showed that it required 10x more computation just to achieve moderate performance. We therefore restrict our experiments to VARC.

3 METHOD

3.1 OBTAINING RULE-INDUCTIVE TEST-TIME TASK EMBEDDINGS

Issues with test embeddings. Hu et al. (2026) show that VARC *train* task embeddings cluster semantically similar tasks and shared (sub-)rules, suggesting that embeddings support implicit rule induction. However, *test* embeddings produced by Full-TTT were not analyzed. In our preliminary experiments, test embeddings exhibit a strong distribution shift from train embeddings, staying often close to their random initialization. As a result, semantic neighborhood structure is lost, weakening rule-induction usefulness at test time. Although this mismatch does not hurt final task-solving performance, it raises efficiency and interpretability concerns: even for duplicates of training tasks presented at test time, the pipeline finetunes 18M parameters per task yet fails to recover the corresponding train embeddings. Moreover, we observed that zeroing out the task embedding causes little performance drop, indicating that the backbone can ignore it. We argue that a desirable model should produce consistent, rule-inductive task embeddings in both training and test phases.

A two-step finetuning approach. Drawing on these observations, we derive a simple two-step TTT approach, shown in Figure 1, that allows learning meaningful test task embeddings while incidentally improving performances. Given a test task, we first freeze the backbone and learn only the task embedding. We denote this step as Embed-TTT. We then freeze that learned embedding and finetune the backbone (Embed-Full-TTT). Embed-TTT enables positioning the test task embedding in the same space as the training embeddings because they share the same backbone, while Embed-Full-TTT allows the model to adjust to the learned embedding to solve the task efficiently.

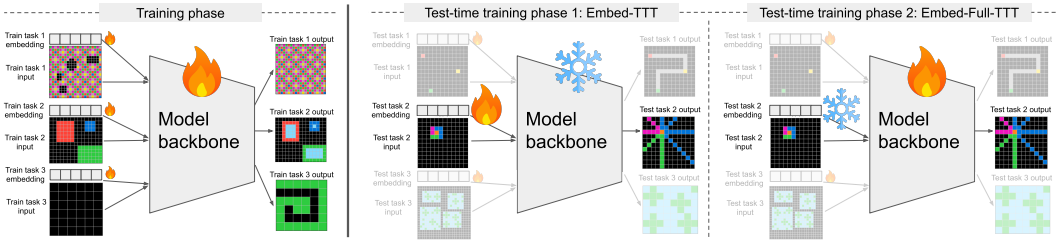

Figure 1: **Method.** After training a model that learns task embeddings that condition the backbone (left), we elicit implicit rule induction at test time: we first freeze the backbone and learn a task embedding (Embed-TTT, center), then freeze the embedding and finetune the backbone (right).

3.2 Evaluating implicit rule induction through task embeddings

Rules unavailable: training embeddings retrieval. In datasets like ARC, where each task is unique, automated evaluation of rule induction largely reduces to checking whether duplicate training tasks retrieve their originally learned embeddings at test time. More precisely, we compute the proportion of tasks for which the original embedding is within the top-k closest embeddings of the newly TTT-produced embedding. Limitations of this approach are the fact that memorization alone might suffice to retrieve the original task embedding, the fact that multiple embeddings might solve the tasks, and the fact that, for training tasks that the model was not able to solve, different embeddings might be better suited than the original one. Consequently, this default protocol should be interpreted with caution: it provides only indirect evidence of meaningful implicit rule induction at test time, mostly reliable when test rules are in-distribution with respect to the training rules distribution.

Rules available: linear probing to rules. When rules (or concept groups) are defined and task generators can produce multiple task instances per rule, in-distribution rule induction can be evaluated more cleanly. The setup mirrors self-supervised learning: learn training embeddings through an auxiliary objective, then assess test embeddings with a linear probe on a downstream target. In our ARC-like setting, each data "unit" is a task, and the auxiliary objective is solving the task by finetuning its task embedding. Given rule labels for training tasks, we collect train embeddings, fit a linear probe from embeddings to rules, and apply it to test embeddings. We probe performance then measure how much rule information is encoded in the embeddings. We distinguish two cases:

- **Non-parametric rules.** Linear probing is a classification problem, and we measure implicit rule induction by the accuracy on the test tasks of the linear classifier learned from the train tasks.
- **Parametric rules.** Linear probing is a linear regression problem, and we measure the root mean squared error (RMSE) between linearly regressed and ground-truth parameters on the test tasks from the linear regressor learned from the train tasks.

4 Results

4.1 Quantitative in-distribution implicit rule induction evaluation

Datasets and experimental details. We conduct the following experiments.

- ARC-1. We start from the trained VARC of Hu et al. (2026) and perform Full-TTT and Embed-TTT on duplicates of the training tasks to measure train embedding retrieval accuracy.
- ConceptARC. We train a VARC model from scratch and perform the same experiment as for ARC-1. We also evaluate the linear probing classification accuracy by considering the 16 concepts underlying the tasks as proxies for the rules that generated them. Since no train-test split exists, we perform a 5-fold cross-validation, each time learning the linear mapping from 8 tasks per concept and evaluating it on the remaining 2 tasks. We report the average accuracy across the runs.
- Custom "LUCD". We design a simple dataset with known rules in the *non-parametric* setting. Each task consists in applying zero, one, or two transforms to a single-cell object positioned randomly on a 10×10 grid, sampled (with repetition enabled) from 4 "atomic" transforms: move one cell to the left (L), up (U), dilate in the cardinal directions (C), dilate in all directions (D). Since all atomic rules commute with each other, this yields 15 different possible rules.

Table 1: **Quantitative implicit rule induction.** We evaluate Embed-TTT and Full-TTT on ARC-like datasets in their ability to produce embeddings that potentially elicit rule induction. It appears that Embed-TTT excels, while Full-TTT generally loses any semantically meaningful information.

Train dataset	Num. tasks train/probe	Evaluation	Metric	Metric score		
				Chance	Full-TTT	**Embed-TTT**
ARC-1 train	400/same	Emb. retrieval (train)	Top-5 match ↑	1.3%	6.5%	**86.25%**
ConceptARC	160/same	Emb. retrieval (train)	Top-5 match ↑	3.1%	69.4%	**98.8%**
ConceptARC	160/same	Lin. classif. (train)	Accuracy ↑	6.2%	21.3%	**60.6%**
Custom LUCD	15/150	Lin. classif. (test)	Accuracy ↑	6.7%	11.3%	**100%**
Custom Moves	121/1210	Lin. regress. (test)	RMSE ↓	-	2.94	**2.12**

- Custom "Moves". We design a simple dataset with known rules in the *parametric* setting. Each task consists in moving a single-cell object on a 10×10 grid by at most five cells both horizontally (left (L), right (R)) and vertically (up (U), down (D)), yielding a total of 121 different rules (for instance, "L4-D3" is the rule moving 4 cells to the left and 3 down, associated to coordinates (-4,-3)).

For "LUCD" (resp. "Moves"), we generate 1 training task and 10 test tasks per rule, with 10 training and 10 test input-output pairs per task(we provide examples in Appendix A (resp. Appendix B)). We evaluate rule induction capabilities on these datasets as described in Section 3.2.

Results. Table 1 shows that, across all evaluations, Embed-TTT largely outperforms the baseline Full-TTT, which performs mostly near chance. Also, we observed that tasks successfully solved are generally better retrieved (91.2% with Embed-TTT on ARC) than failed tasks (77.1%). For ConceptARC, the concepts are only a meta-category associated with the tasks and do not fully characterize the task-generating process. Hence, observing a perfect mapped accuracy is likely out of reach, but Embed-TTT performs reasonably well, with 60.6% accuracy. In LUCD, one-shot learning a linear classifier from a single task instance per rule yields a perfect test accuracy, whereas Full-TTT performs at chance. This is a strong sign that Embed-TTT is capable of implicit rule induction, since test tasks were never seen during training (contrary to the previous evaluations). We observe the same trend for our regression evaluation on Moves. We used the cosine distance to determine embeddings neighbors. In addition, we can align the probing setup with each method's own test-time dynamics: we first run each TTT protocol on duplicates of the training tasks, then learn the linear mapping from these TTT-produced train embeddings (instead of from frozen post-training embeddings), and finally apply it to test embeddings. Here, on LUCD, Full-TTT rises to 39.2% while Embed-TTT remains at 100%; on Moves, Full-TTT improves RMSE to 1.27 and Embed-TTT to 0.44.

Figure 2 compares UMAP projections of Full-TTT and original train embeddings with projections of Embed-TTT and original train embeddings, showing a clear distribution shift in the former case while Embed-TTT better recaptures the structure of the manifold of the original embeddings. The ground-truth embedding is generally among the closest neighbors (computed in the original embedding space) of the "test" Embed-TTT-produced embedding under consideration (most points are dark green), while Full-TTT embeddings are generally much further away from the original embeddings. To assess whether the local geometry of ARC-1 task embeddings is preserved (which could happen regardless of retrieval performance), we compared k-nearest-neighbor structure between the original train embedding space and embeddings from Embed-TTT and Full-TTT. For each task, we computed top-k neighbors in each space (excluding self), then measured neighborhood agreement with the original space via set overlap proportion. We found a large and consistent gap in favor of Embed-TTT: mean overlap at $k = 10$ is 33% for Embed-TTT vs 4% for Full-TTT (with similar observations across $k \in \{1...50\}$). These results indicate that Embed-TTT preserves local relational structure from the pre-trained task-embedding manifold, whereas Full-TTT distorts it.

4.2 Qualitative "out-of-distribution" evaluation of implicit rule induction

Dataset. ARC-1 provides 400 evaluation/test tasks, that can differ from the training tasks rule-wise, making them potentially "out-of-distribution" (OOD). We can thus qualitatively examine the similarity between test and training tasks to measure how well an embedding-producing method works in this OOD case. This analysis is not feasible with ConceptARC due to the absence of a test set, and not relevant for our custom tasks since we generated only simple in-distribution tasks.

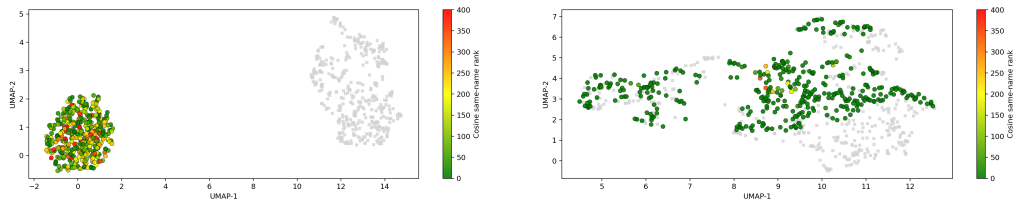

Figure 2: **UMAP projections for ARC-1.** Full-TTT embeddings (left) are projected far from the original embedding distribution, while Embed-TTT embeddings (right) are projected adequately onto it. The original embedding usually ranks among the closest train embeddings to the fine-tuned one with Embed-TTT (most points are dark green, indicating a good rank).

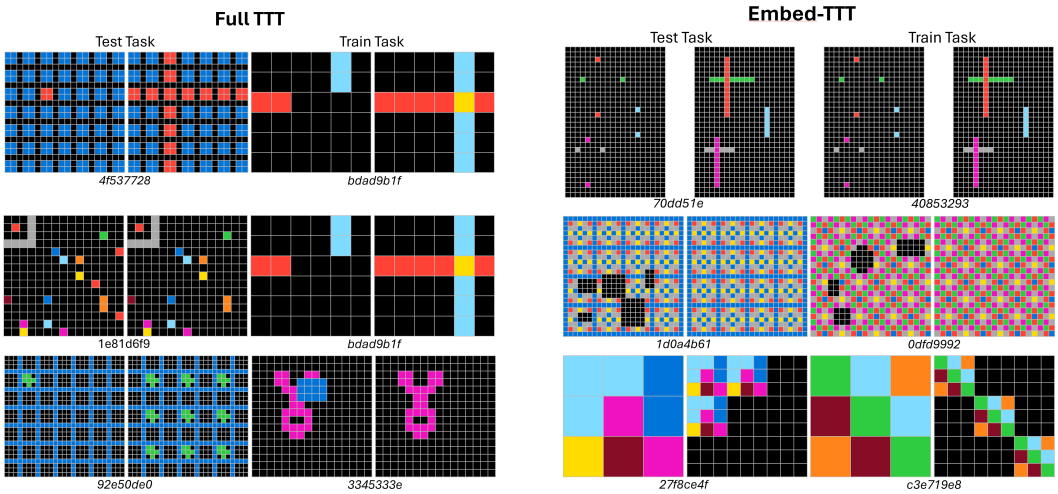

Figure 3: **Qualitative evaluation through test-train task similarity.** For Full-TTT (left part) and Embed-TTT (right part), we report the closest associations between test (left input-output pairs per part) and train (right input-output pairs) tasks. Full-TTT does not associate tasks that seem to share common rules, while Embed-TTT associates tasks that are clearly similar rule-wise, the top one being a task duplicate with different IDs (070dd51e and 40853293) in test and train sets.

Results. For that purpose, we search for the closest train task in the embedding space for each test task. To mimic the optimization dynamic undergone by the test task in this process, the search occurs among the train embeddings obtained after Embed-TTT was performed, so that embeddings obtained following the same process are compared. Figure 3 shows the closest test-train task associations from Full-TTT (left) and Embed-TTT (right). We observe that Full-TTT does not find relevant task associations, likely because of the distribution shift between test and train embeddings is too large, as already seen in Figure 2 (more are shown in Appendix C). On the other hand, Embed-TTT yields test-train associations that are relevant, which is non-trivial despite the good in-distribution results. In particular, the first association is an exact task duplicate shared between the test and the train sets of ARC-1. The other tasks shown clearly display the same rule semantics, showing a regular inpainting task and a copy task according to the dominant color of the input grid. In our observations, close associations are usually relevant, while farther ones are less so, as expected given that some ARC-1 test tasks differ radically from the training tasks. We did not quantify this observation further, as it is relatively subjective. Still, qualitative examination shows that Embed-TTT enables meaningful comparison between test and train tasks, which is not the case with the default Full-TTT approach.

4.3 Cross-dataset capabilities

Results. By acting as an implicit rule inductor, Embed-TTT can be used to test whether a model trained on an ARC-like dataset can induce and execute rules underlying tasks of another ARC-like dataset. In that regard, Table 2 compares the performances of the original VARC ViT model with

Table 2: **Cross-dataset results.** Our Embed-Full-TTT allows to reach better final per-task performances than Full-TTT on ARC-1, ConceptARC and Mini-ARC. In addition, Embed-TTT, solves a significant amount of tasks by just finetuning the embeddings, indicating that rules underlying test tasks might not always differ much from train tasks. Pair-level (instead of task-level) evaluation on ConceptARC(*) shows a large gap, indicating difficulties in solving all pairs of a given task consistently. Finally, ConceptARC concepts inferred on ARC-1 tasks by task embeddings similarity decently match manual annotations despite the model was never trained for concept retrieval.

Train dataset	Test dataset	Pass@1 per TTT method			Pass@2 per TTT method		
		Full	Embed	Embed-Full	Full	Embed	Embed-Full
ARC-1 train	ARC-1 test	48.8%	13.0%	**49.5%**	53.8%	17.2%	**55.2%**
ARC-1 train	ConceptARC	23.8%	8.8%	**28.1%**	32.5%	14.4%	**37.5%**
ARC-1 train	Mini-ARC	59.7%	22.2%	**63.8%**	62.4%	34.9%	**71.1%**
ARC-1 train	ConceptARC (*)	47.1%	24.0%	**49.8%**	53.8%	33.3%	**57.1%**
Concept retrieval in ARC		Top-1: Test 36.3% (29/80), Train 35.0% (28/80), from Embed-TTT					

Full-TTT, Embed-TTT, and Embed-Full-TTT on ARC-1, ConceptARC and Mini-ARC (Kim et al., 2022) tasks. We report the proportion of test tasks solved at pass 1 and pass 2 in the test datasets.

We observe that Embed-Full-TTT—finetuning the task embedding first, freezing it, then finetuning the model backbone—provides a moderate performance improvement on ARC-1 test set over the default Full-TTT and a larger improvement on ConceptARC and Mini-ARC. These improvements might be due to a better conditioning on the backbone finetuning based on an improved embedding. Interestingly, Embed-TTT, by finetuning only the task embeddings at test time, which accounts for $< 0.01\%$ of the model weights, suffices to score more than 13% on ARC-1 at pass 1 (resp. 17% pass 2), 8% (resp. 14%) on ConceptARC, and 22% (resp. 34%) on Mini-ARC. This indicates that the VARC model has learned general rule induction and execution capabilities, but also that some ARC-1 test tasks, ConceptARC and Mini-ARC tasks might be similar to the ARC-1 training tasks. These percentages also represent a significant fraction of what can be achieved by the model when the backbone finetuning is allowed. From an efficiency perspective measured as weights to finetune, Embed-TTT results are exceptionally high and indicate, to some extent, the generalizability of our approach beyond just ARC-1 tasks. As a point of comparison, we tested two recent open-weight language models on the same public ARC-1 test set with the same pass@1 evaluation as used in our study: Llama-4-Maverick scores only 4.2% and Kimi K2.5 scores 27.8%. These results show that our approach outperforms these models on the same controlled settings. Proprietary models used in the official ARC leaderboard are evaluated on a private dataset and thus might have been trained on the test set considered here, hence we refrain from running expensive and potentially flawed comparisons, while acknowledging that such models are probably substantially more capable.

Let us note that Table 2 also shows that, for ConceptARC, independent performance across test input-output pairs (480) is significantly higher than when aggregated per task (3 test pairs per task). This implies that the model does not solve ConceptARC tasks on a consistent basis, with successful test samples scattered across the tasks rather than consistently succeeding or failing on specific tasks. For ARC, a similar analysis gives a boost of only 0.3% across all metrics, mainly because most test tasks contain only one test sample, making the evaluation almost equivalent in both cases.

Further Embed-TTT experiments on LUCD and Moves, and broader and deeper test-time search for ARC-1, are reported in Appendix D. They show that, curiously, task-embedding finetuning alone may fail even on simple OOD transformations, while expanding the search yields modest additional gains.

Concept retrieval in ARC. In addition, given that they were obtained by using the same backbone, ARC-1 and ConceptARC embeddings lie in comparable regions of the embedding space (akin to Figure 2 (right)). Hence, we can compare them, by assigning each ARC-1 task its closest ConceptARC task, using Embed-TTT-produced embeddings for ConceptARC tasks, ARC-1 train and ARC-1 test tasks. In particular, we want to know if the core concepts close to ARC-1 tasks are indeed related to the tasks. For that purpose, we manually annotated a balanced subset of 80 ARC-1 train tasks and 80 ARC-1 test tasks (5 for each of the 16 concepts uniformly spread across cosine distances), by indicating which concepts (at most 3) were best related to the tasks. We then check if the closest

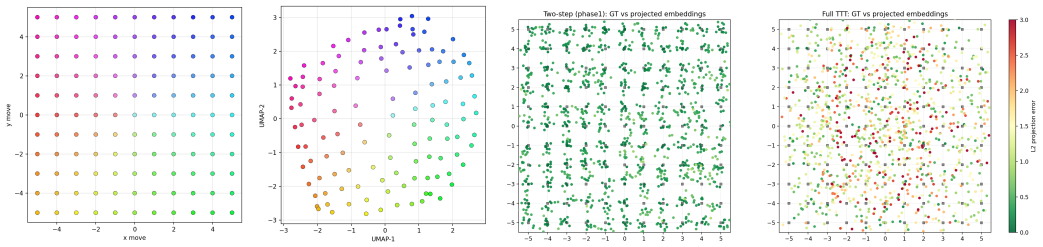

Figure 4: **Embeddings structure.** In our Moves dataset, the task geometry (left) is reflected by the UMAP of train embeddings (center-left), which forms an almost regular grid. We then learn a linear map from train embeddings finetuned with Embed-TTT (resp. Full-TTT) to their ground-truth move coordinates (gray squares) (center-right, resp. right), and apply it to test embeddings (points colored by projection error). Embed-TTT preserves a regular grid with low error, unlike Full-TTT.

concept found matches one of the annotations. The results are reported in Table 2 and show decent agreement: 36.3% on test and 35.0% on train ARC-1 tasks. Importantly, the agreement is stronger for closer embedding associations: in both train and test splits, the closest-distance quartile yields the highest agreement (60% on test, 50% on train). Overall, nearest ConceptARC neighbors provide a useful interpretability signal, especially at small distances, but do not suffice as concept classifiers for ARC-1 tasks. However, we noticed that many ARC-1 tasks are not perfectly aligned with the predefined concepts, and that additional concepts would improve coverage. Still, the results are far above pure chance (about 6.3%). Hence, Embed-TTT enables, for the first time in this setting, a practical examination of meaningful connections between different ARC-like datasets.

4.4 Compositional Structure of the Embeddings

Datasets and scope. We examine how the model structures the Moves embedding space through the cosine similarity of composed transforms (e.g. R3-U2) to their atomic transforms (R3-U0, R0-U2). Then, we investigate which tasks Embed-TTT solves from subsets of our initial training dataset.

Compositions vs. atoms. Across the 100 composed transforms, we find that the two atoms are both the most or second-most similar purely horizontal and vertical atoms for 94% of the cases. The cosine similarities with "unit" transforms (in the example case R1-U0 and R0-U1) are much weaker, indicating a lack of scaling capability, in favor of a plain recombination of individual "already-scaled" atomic projections. This dataset enables an intuitive visualization: Figure 4 (left and center-left) shows that the UMAP projection of the Embed-TTT embeddings follows an almost regular grid-like pattern that mimics the geometry of the dataset, where each task of the dataset can be associated to its move coordinates. Similar visualizations from PCA are provided in Appendix E. Further, a linear mapping learned from Embed-TTT train embeddings efficiently projects test embeddings onto the grid, contrary to a mapping learned from Full-TTT train embeddings (right-center and right panels).

Embed-TTT allows interpolation between tasks. For the Custom Moves dataset, we further investigate which moves might be sufficient as training set so that the model generalizes to the remaining unseen moves. In that regard, Figure 5 (left) shows various configurations, where training moves are represented in gray squares while test moves are circles colored according to the model performance on these moves with Embed-TTT. The top left panel indicates that interpolation inside known moves is achievable at test time, while extrapolation outside known moves is not possible in that configuration. The top right panel shows, however, that, even if the model was trained on all move types in each direction, it is not able to compose them without having seen any composition beforehand. The bottom left panel shows 30 randomly selected train moves and indicates that test moves located on lines where only one training move was present are less easily achievable. Finally, the bottom right panel shows that, when 3 training moves are present on each row and each column (33 train moves in total), almost all the other moves can be solved at test time by Embed-TTT, at the exception in this case of the task involving no move, likely because the model was trained to always do at least one directional transform. With comparable number of train moves (30 vs 33), this hints that train moves configuration can play a major role into generalization capabilities.

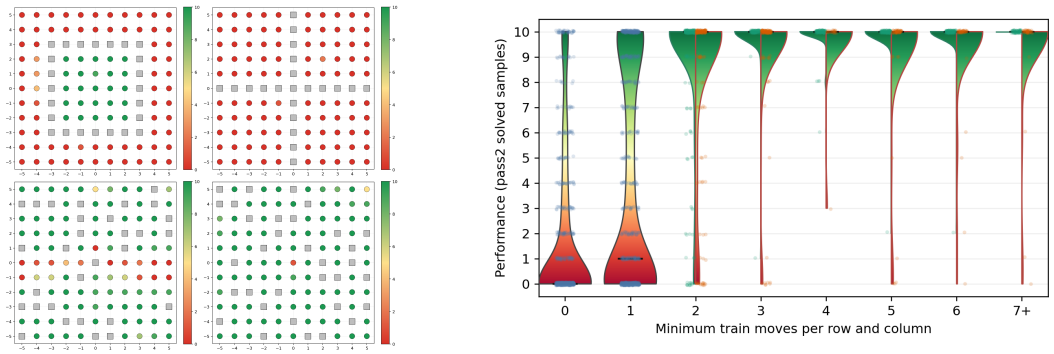

Figure 5: **Task inter/extrapolation.** (Left) Visualization of Moves tasks. Gray squares denote train tasks; circles denote test tasks, colored by success under Embed-TTT finetuning. Test tasks are mostly solved when at least 2 train moves lie on both their row and column, suggesting interpolation is feasible while extrapolation is limited. (Right) Violin plots aggregate test successes by the minimum number of train tasks on the same row/column. A clear transition appears at 2 ("grokking" effect). Split violins show "inside" test tasks (left half) are better solved than "outside" tasks (right half).

These observations suggest that the minimal number of training moves present in the same rows and columns as the test moves is a strong predictor of success. Concretely, we can group test task performances by the minimum value between the number of training tasks on their row and on their column (from 0 to 10, shown in Figure 5, right). A clear transition appears at 2: the model tends to "grok" once at least 2 aligned training moves are available. A plausible explanation is that reusing embedding dimensions for a given direction requires repeated evidence; with only one instance, there is little pressure to factorize shared structure. The violin plots are further split into two parts vertically. The left-hand side corresponds to test moves that are "inside" the (at least) two training moves located on the same row or column, the right-hand side for "outside" test moves. Test moves located "inside" train moves are more easily solved, yet they do not correspond to extrapolation capabilities, since they still lie within the convex hull of the training moves. Further, when adding a color change operation, even when no compositions are shown, Embed-TTT enables the correct projection to a point that lies closest to both valid moves and color changes, whereas Full-TTT fails completely (see Appendix F).

We note a limitation in that Embed-TTT cannot "add moves". For instance, in the top left panel of Figure 5, the model learned to move +1 to the right (through the train moves (+1,+3) and (+1,-3)) and obviously by +3. However, we found no way (heuristically or by constrained optimization) to create an embedding that activates both +3 and +1 to achieve +4. The issue likely lies in the execution of the embedding, because the backbone was never trained on such combinations. We also found that forcing the model to apply two moves iteratively that could be optimized jointly was unsuccessful. The lack of prior training on such iterative composition likely causes this failure.

5 Conclusion

We introduced Embed-TTT, a test-time training protocol for Vision ARC (VARC) that first optimizes task embeddings to solve ARC-like tasks, before an optional backbone finetuning. Across ARC-1, ConceptARC, and controlled datasets with known rules, Embed-TTT yields task embeddings that are more semantically meaningful and substantially more rule-aligned than those from the original full finetuning protocol. We show that these embeddings enable test-train task associations, cross-dataset task comparisons, and task geometry recovery. As limitations, Embed-TTT requires iterative test-time optimization through the full model, and the two-step process is slower to reach only slightly better performances. However, the value of Embed-TTT embeddings resides mainly in their rule induction capabilities, supported by our results and to be distinguished from rule execution by the backbone.

Our findings also convey a broader message for evaluating ARC-like benchmarks: final accuracy alone does not suffice to characterize reasoning. More focus on assessing rule induction is needed, with a rigorous separation between in-distribution and out-of-distribution at the rule level. Eventually this will lead to more reliable claims about model "reasoning" capabilities.

AI USE STATEMENT

(This section is **required** and does not count toward the page limit.)

In this work, we used generative AI tools for generating the Moves and LUCD datasets, formatting ARC-like datasets in the ARC-1 format, assist with translation or rewriting of some paragraphs, write the code for the analysis of the results. We have not used generative AI tools for proposing or refining hypotheses, design or provide feedback on research methodology or experiments, implement methods, support qualitative and thematic data analysis, interpret results; developing theoretical models or conceptual frameworks, formulating mathematical claims or proving them, are not applicable to this work. Additionally, we used generative AI tools for creating templates to design the figures, editing software code (to manage experiments on GPU clusters). We have reviewed all AI-assisted work: we checked the generated datasets, we ensured formatting allowed further seemless processing, we reviewed and revised all the text of the main document, checked the validy of the results through manual inspection of most of them. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

(This section is **recommended** and does not count toward the page limit.)

Not applicable.

REPRODUCIBILITY STATEMENT

(This section is **recommended** and does not count toward the page limit.)

Our Embed-TTT procedure and its evaluation protocols are described in Section 3. The appendix describes the controlled LUCD and Moves datasets, and provides additional retrieval, OOD generalization, test-time search, PCA, and compositional analyses. An anonymized supplementary material contains the modified VARC data loader, the offline-training and two-step test-time-training implementations, and all datasets used in our experiments. The supplementary material also documents the relevant preprocessing, task splits, execution protocol, and result-computation procedures, enabling independent reproduction and verification of the reported results.

REFERENCES

Ekin Akyürek, Mehul Damani, Adam Zweiger, Linlu Qiu, Han Guo, Jyothish Pari, Yoon Kim, and Jacob Andreas. The Surprising Effectiveness of Test-Time Training for Few-Shot Learning. In *International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 942–963, Vancouver, Canada, 2025. ML Research Press. URL https://proceedings.mlr.press/v267/akyurek25a.html.

Claas Beger, Ryan Yi, Shuhao Fu, Kaleda Denton, Arseny Moskvichev, Sarah W. Tsai, Sivasankaran Rajamanickam, and Melanie Mitchell. Do AI Models Perform Human-like Abstract Reasoning Across Modalities? *arXiv*, abs/2510.02125, 2025. doi: 10.48550/arXiv.2510.02125. URL https://doi.org/10.48550/arXiv.2510.02125.

François Chollet. On the Measure of Intelligence. *arXiv*, abs/1911.01547, 2019. doi: 10.48550/arXiv.1911.01547. URL https://doi.org/10.48550/arXiv.1911.01547.

François Chollet, Mike Knoop, Gregory Kamradt, and Bryan Landers. ARC Prize 2024: Technical Report. *arXiv*, abs/2412.04604, 2024. doi: 10.48550/arXiv.2412.04604. URL https://doi.org/10.48550/arXiv.2412.04604.

Felix Hill, Adam Santoro, David G. T. Barrett, Ari S. Morcos, and Timothy Lillicrap. Learning to Make Analogies by Contrasting Abstract Relational Structure. In *International Conference on Learning Representations (ICLR)*, pp. 1–18, New Orleans, Louisiana, USA, May 2019. URL https://openreview.net/forum?id=SylLYsCcFm.

Michael Hodel. Addressing the Abstraction and Reasoning Corpus via Procedural Example Generation. *arXiv*, abs/2404.07353, 2024. doi: 10.48550/arXiv.2404.07353. URL https://doi.org/10.48550/arXiv.2404.07353.

Keya Hu, Ali Cy, Linlu Qiu, Xiaoman Delores Ding, Runqian Wang, Yeyin Eva Zhu, Jacob Andreas, and Kaiming He. ARC Is a Vision Problem! In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2537–2546, Denver, Colorado, USA, June 2026. IEEE.

Alexia Jolicoeur-Martineau. Less is More: Recursive Reasoning with Tiny Networks, 2025. URL https://arxiv.org/abs/2510.04871.

Subin Kim, Prin Phunyaphibarn, Donghyun Ahn, and Sundong Kim. Playgrounds for Abstraction and Reasoning. In *NeurIPS 2022 Workshop on Neuro Causal and Symbolic AI (nCSI)*, pp. 1–6, Virtual conference, November 2022. URL https://nips.cc/virtual/2022/57706.

Brenden M. Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2873–2882. ML Research Press, July 2018. URL https://proceedings.mlr.press/v80/lake18a.html.

Cassidy Langenfeld, Claas Beger, Gloria Geng, Wasu Top Piriyakulkij, Keya Hu, Yewen Pu, and Kevin Ellis. Bongards at the Boundary of Perception and Reasoning: Programs or Language? *arXiv*, abs/2602.03038, 2026. doi: 10.48550/arXiv.2602.03038. URL https://doi.org/10.48550/arXiv.2602.03038.

Martha Lewis and Melanie Mitchell. Evaluating the Robustness of Analogical Reasoning in Large Language Models. *Transactions on Machine Learning Research*, 2:1–33, 2025. URL https://openreview.net/forum?id=t5cy5v9wph.

Wen-Ding Li, Keya Hu, Carter Larsen, Yuqing Wu, Simon Alford, Caleb Woo, Spencer M. Dunn, Hao Tang, Michelangelo Naim, Dat Nguyen, Wei-Long Zheng, Zenna Tavares, Yewen Pu, and Kevin Ellis. Combining Induction and Transduction for Abstract Reasoning. In *International Conference on Learning Representations (ICLR)*, pp. 1–44, Singapore, April 2025. URL https://openreview.net/forum?id=UmdotAAVDe.

Jacqueline R. M. A. Maasch, John Kalantari, and Kia Khezeli. CausalARC: Abstract Reasoning with Causal World Models. In *Advances in Neural Information Processing Systems (NeurIPS) Workshop*, volume 39, pp. 1–28, San Diego, California, USA, December 2025. Curran Associates, Inc. URL https://openreview.net/forum?id=jO5pG9uRez.

Patrick J. Mineault, Thomas L. Griffiths, and Sean Escola. Cognitive Dark Matter: Measuring What AI Misses. *arXiv*, abs/2603.03414, 2026. doi: 10.48550/arXiv.2603.03414. URL https://doi.org/10.48550/arXiv.2603.03414.

Arsenii Kirillovich Moskvichev, Victor Vikram Odouard, and Mitchell Mitchell. The ConceptARC Benchmark: Evaluating Understanding and Generalization in the ARC Domain. *Transactions on Machine Learning Research*, 8:1–20, 2023. URL https://openreview.net/forum?id=8ykyGbtt2q.

Wasu Top Piriyakulkij, Yichao Liang, Hao Tang, Adrian Weller, Marta Kryven, and Kevin Ellis. PoE-World: Compositional World Modeling with Products of Programmatic Experts. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 39, pp. 1–30, San Diego, California, USA, December 2025. Curran Associates, Inc. URL https://openreview.net/forum?id=obwRcksFZw.

Wen-Jie Shu, Xuerui Qiu, Rui-Jie Zhu, Harold Haodong Chen, Yexin Liu, and Harry Yang. LoopViT: Scaling Visual ARC with Looped Transformers, 2026. URL https://arxiv.org/abs/2602.02156.

Eric Todd, Millicent Li, Arnab Sen Sharma, Aaron Mueller, Byron Wallace, and David Bau. Function Vectors in Large Language Models. In *International Conference on Learning Representations (ICLR)*, pp. 17282–17333, Vienna, Austria, May

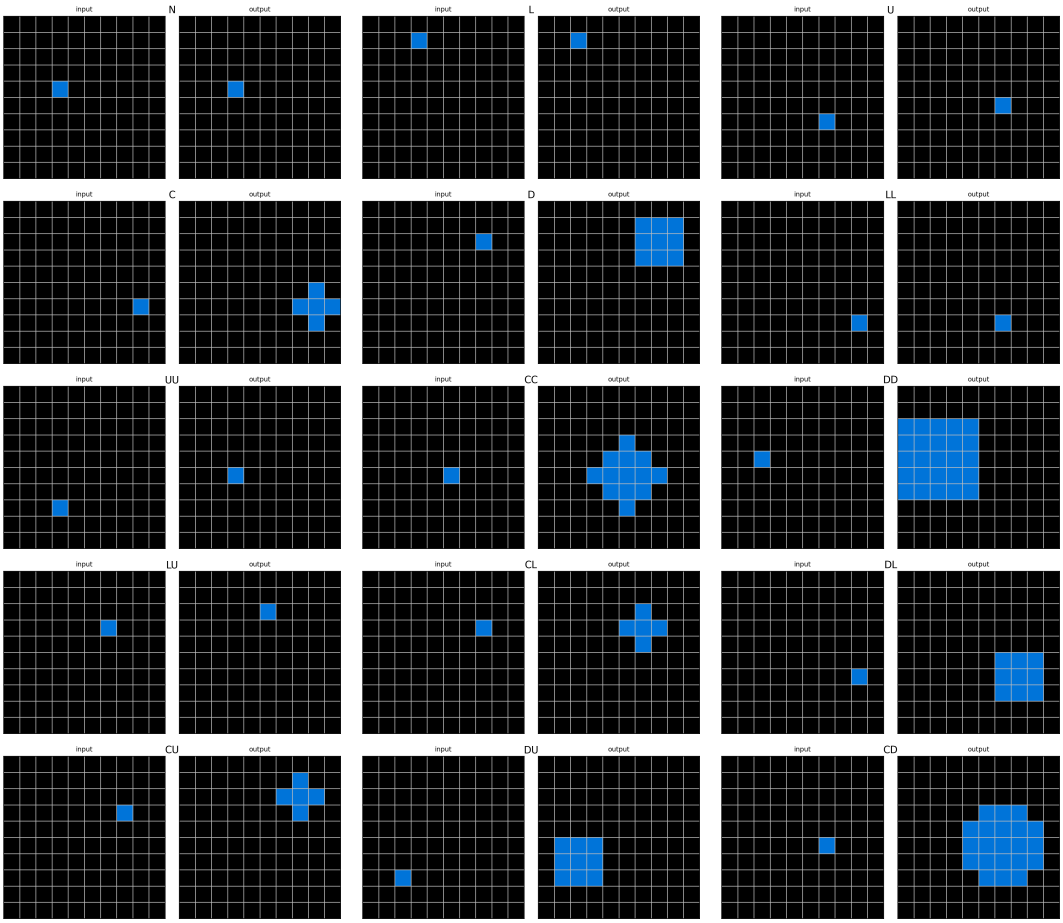

Figure 6: **Our LUCD tasks**. Examples of input-output pairs per LUCD task of our dataset. Tasks are encoded as 'left' (L), 'up' (U), 'cross' (C), 'dilate' (D), 'neutral' (N) transforms, composed with each other.

2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/4ae163cb8788970e53b4fd9578141139-Abstract-Conference.html.

Karel Veldkamp, Claire Stevenson, Hannes Rosenbusch, and Luca Thoms. Solving ARC visual analogies with neural embeddings and vector arithmetic: A generalized method. *OSF*, 2023. doi: 10.17605/OSF.IO/AKP86. URL https://osf.io/akp86/.

A Our Custom LUCD dataset

One input-output pair for each of our 15 tasks of our custom LUCD dataset are represented in Figure 6.

B Our Custom Moves dataset

Some input-output pairs of our Moves dataset are represented in Figure 7.

C Qualitative similar tasks retrieval

Additional results of the retrieval of train tasks 'similar' to test tasks according to Full-TTT and Embed-TTT are shown in Figure 8.

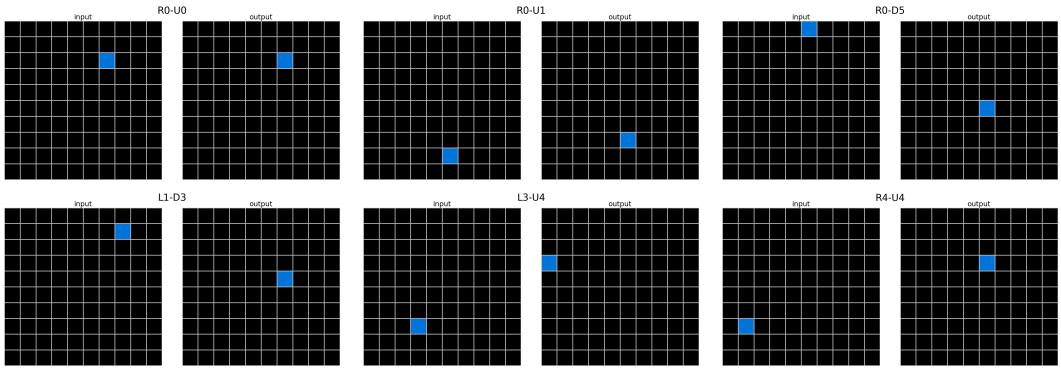

Figure 7: **Our Moves tasks.** Examples of input-output pairs per Moves task of our dataset. Tasks are encoded as left (L) or right (R) moves and up (U) and down (D) moves.

D Additional Embed-TTT experiments on OOD Generalization and Test-Time Search

Interestingly, even though VARC is a highly capable model trained on many ARC tasks and extra RE-ARC (Hodel, 2024) samples, it does not necessarily learn the simple transformations involved in LUCD or Moves. Indeed, with Embed-TTT on LUCD, pass@1 is 56% and pass@2 is 67.3%, on Moves, pass@1 is 11.6% and pass@2 is 19.8% (and of course any full finetuning reaches 100% but the expectation was to reach that with just task embedding finetuning). This indicates that models trained on some ARC-like tasks primarily "interpolate" between the contained functionalities, while even simple OOD tasks might not be directly solved by just finetuning the task embedding. This point is further reinforced in section 4.4. This is a curious property, and points to how brittle models might be even when they appear strong on benchmarks claiming to require OOD generalization.

Additional experiments with Embed-TTT on ARC-1 test set showed that performance could be pushed slightly further. By increasing the "breadth" of search by finetuning up to 8 different task embeddings at test time (instead of just one) and then using the one with the lowest training loss, pass@1 (resp. pass@2) reaches 14.8% (resp. 18.5%). Further increasing the "depth" of search by finetuning two task embeddings applied sequentially allows to reach up to 17.3% (resp. 20.3%). Such results show that the trained model still contains further capabilities, yet the gains of continuing to further increase breadth and depth are likely marginal and would be computationally expensive. Hence, we did not investigate this direction deeper.

E PCA results on Moves

Embed-TTT embeddings capture well vertical and horizontal ground-truth moves, as shown in Figure 9. In comparison, Full-TTT mostly captures the left-right and up-down distinctions, but not a fine regular ordering of the moves, as shown in Figure 10. The PCA values also tend to indicate a much more condensed projection.

F Experimenting with moves and color switch

We designed a controlled ARC-like benchmark where each task applies (i) a spatial move and (ii) a color switch to a single blue cell. Spatial moves span displacements from -3 to $+3$ on each axis ($7 \times 7 = 49$ move rules), and color switches span 8 target colors, yielding $49 \times 8 = 392$ task types. The training split contains only *partial* rules: *move-only* tasks (all moves with no switch) and *switch-only* tasks (all switches with no move), totaling 57 tasks. Test tasks are the remaining move and switch compositions. This setup directly evaluates whether a model that learned "composition" in a training setting (here, composition over moves) can transfer the "composition" ability to an unseen composition at test time (here, composition of moves and switches).

Table 3: Top-k retrieval of both correct constituents (train move and train switch) for composed test tasks. Note that the entire dataset consists of k=57 tasks.

TTT method	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 10$	$k = 30$	$k = 50$	$k = 56$
Embed-TTT	82.81%	91.41%	95.31%	98.18%	100.00%	100.00%	100.00%	100.00%
Full-TTT	0.00%	0.00%	0.00%	0.00%	0.00%	1.30%	35.94%	93.49%

Main observation. At sample level, Embed-TTT without model training solves 0% of composed test tasks, while Full-TTT reaches near-perfect performance ($\approx 99.95\%$) due to full backbone adaptation on easy tasks. However, nearest-neighbor analyses in embedding space show the opposite trend for implicit rule induction: Embed-TTT embeddings consistently retrieve the correct atomic constituents (move and switch) of the test compositions, whereas Full-TTT largely fails to do so.

Top-k dual-components retrieval. To quantify this effect, for each composed test task we check whether *both* correct train move (without switch) and train switch (without move) are present in top-k nearest neighbors. As shown in Table 3, Embed-TTT reaches 100% by $k = 10$ (even by $k = 7$), while Full-TTT remains near zero for a wide range of k and only increases at very large k. This indicates that Embed-TTT preserves strong semantic locality for compositional rule components, whereas Full-TTT does not. In addition, for each composed test task, we also measured: (i) whether the closest "switch" train neighbor is the correct switch, and (ii) whether the closest "move" train neighbor is the correct move. Embed-TTT obtains 100% and 99.5%, respectively; Full-TTT obtains 13.3% (near random chance 12.5%) and 40%.

Interpretation. These results support a dissociation between *rule induction* and *rule execution*: Embed-TTT captures the intended compositional semantics in embedding space, but fails at execution because the frozen backbone was never trained to jointly realize move and switch transformations. This suggests that the bottleneck is primarily executability, not task-level rule identification. On the other hand, Full-TTT unsurprisingly allows to solve the tasks, but at the cost of losing rule induction capabilities.

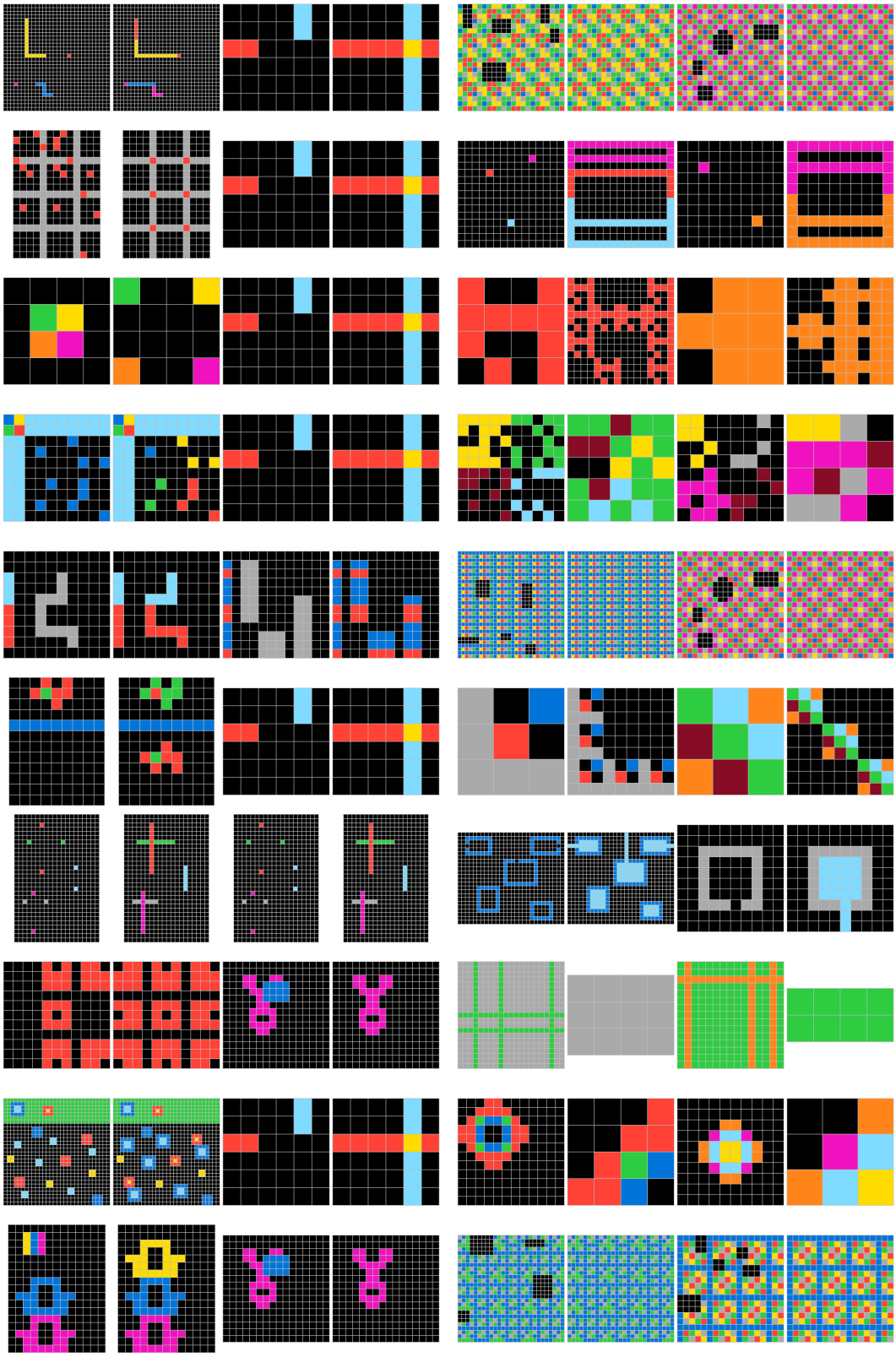

Figure 8: **Similarity retrieval.** (Left) Full-TTT retrieves test-train pairs of tasks that do not seem semantically similar. (Right) Embed-TTT retrieves test-train pairs of tasks that share some common rules.

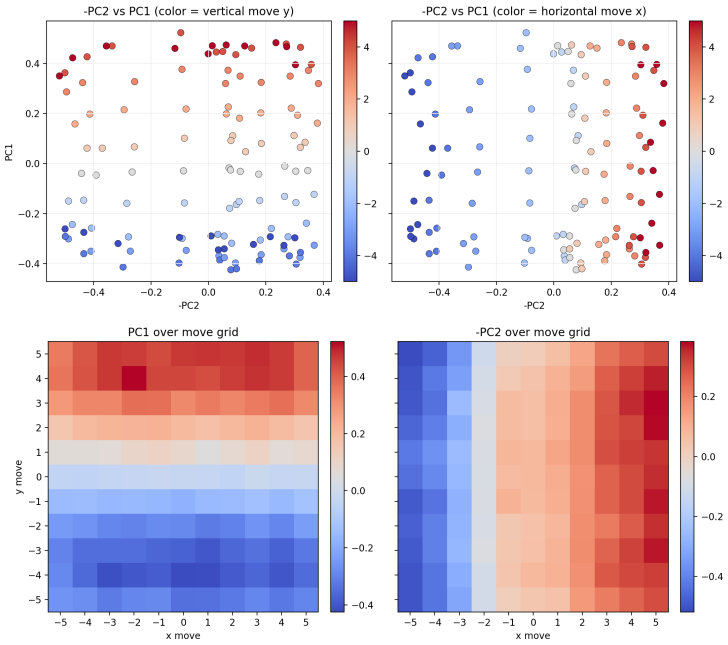

Figure 9: **Embed-TTT PCA on our Moves tasks.** (Top) PCA visualization of the Embed-TTT embeddings of our Moves tasks, colored according to ground truth vertical (left) and horizontal (right) moves. (Bottom) Heatmap showing, for each ground-truth move (represented as a square at (x,y) coordinates), its two main PCA values. These two equivalent plots show that the first component (PCA1) captures vertical moves, the second component (PCA2) captures horizontal moves.

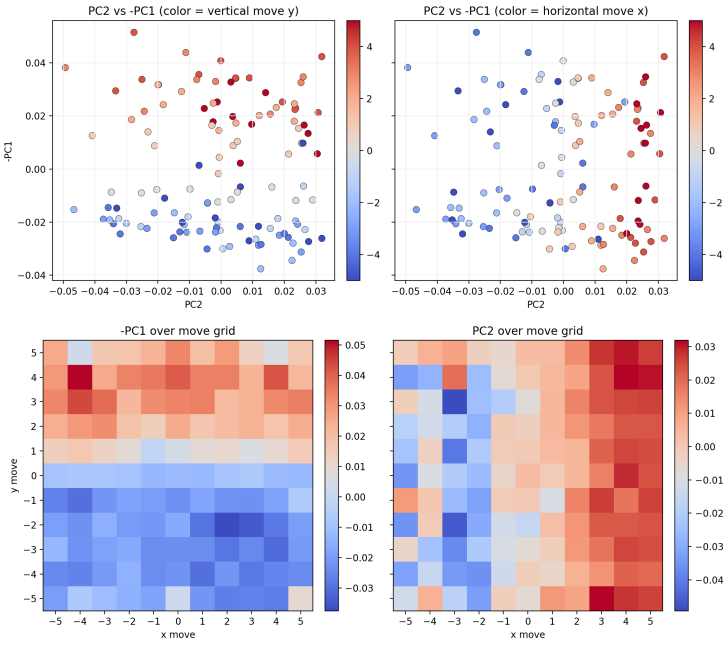

Figure 10: **Full-TTT PCA on our Moves tasks.** (Top) PCA visualization of the Full-TTT embeddings of our Moves tasks, colored according to ground truth vertical (left) and horizontal (right) moves. (Bottom) Heatmap showing, for each ground-truth move (represented as a square at (x,y) coordinates), its two main PCA values. Full-TTT seems to distinguish left moves from right moves, and up moves from down moves, but the exact move ordering seems much less recaptured than with Embed-TTT.