ESANN 2026 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 22-24 April 2026, i6doc.com publ., ISBN 9782875870964.
Available from http://www.i6doc.com/en/.

NSA: Neuro-symbolic ARC Challenge

Pawel Batorski [1] and Jannik Brinkmann[2] and Paul Swoboda [1]

1- Heinrich Heine Universität, Düsseldorf, Germany
2- University of Mannheim, Mannheim, Germany

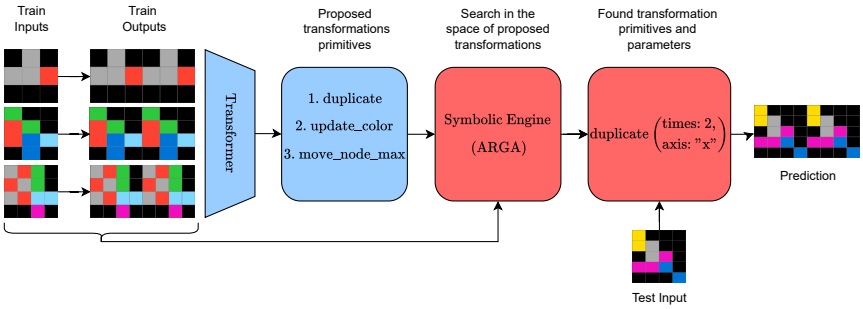

Fig. 1: An overview of NSA: given training input–output pairs, a transformer proposes candidate transformation primitives, which the ARGA [6] symbolic search composes into a program whose resulting transformation is applied to the test input to produce the final prediction.

Abstract. The Abstraction and Reasoning Corpus (ARC) evaluates general reasoning skills that remain challenging for both machine learning models and combinatorial search. We propose NSA, a neuro-symbolic approach that couples a small 25.3M-parameter transformer for proposing DSL primitives with a symbolic program search. The transformer narrows the search space by suggesting promising transformations, is pre-trained on synthetically generated tasks, and is further adapted at test time using task-specific synthetic data. NSA surpasses comparable state of the art on the ARC evaluation set by 27% and compares favourably on the ARC train set. We make our code and dataset publicly available at: `https://github.com/Batorskq/NSA`.

1 Introduction

The Abstraction and Reasoning Corpus (ARC) benchmark [2] is a few-shot testbed for visual reasoning, where each task consists of a small set of input–output grid pairs. Despite rapid progress in large language models (LLMs), current general-purpose systems still perform far below humans on ARC under reasonable compute budgets [9]. In contrast, human subjects solve around 80% of the tasks on average.
Specialized symbolic approaches based on domain-specific languages (DSLs) [3, 8, 6] can in principle describe many ARC transformations, but they suffer from

ESANN 2026 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 22-24 April 2026, i6doc.com publ., ISBN 9782875870964. Available from http://www.i6doc.com/en/.

combinatorial explosion: even moderately richer DSLs lead to search spaces that become intractable within the typical time budget.
These observations suggest combining the strengths of both worlds: DSLs provide the right level of abstraction for object, and relation centric reasoning, while learned components can quickly propose promising transformations instead of exhaustively exploring the entire search space. We follow this neuro-symbolic direction and introduce NSA, a framework that couples an extended ARC DSL with a small transformer-based proposal model.
Our approach builds on the ARGA DSL [6] and extends it with additional transformation primitives (ARGAe), increasing its representational coverage. A transformer with only 25.3M parameters predicts which primitives are likely involved in a task, thereby constraining the symbolic search. Because the ARC training set is limited (400 tasks), we synthetically generate large numbers of training tasks for pre-training and further refine the model at test time by task-specific fine-tuning on synthetically generated examples. Empirically, this neuro-symbolic combination outperforms prior DSL-based and DSL+LLM methods on the ARC evaluation set while using a much smaller neural component.

Contributions In summary, we:

- extend the ARGA DSL with additional generic primitives (ARGAe), increasing the number of ARC tasks that can be represented;
- introduce transformer-guided neuro-symbolic search, where a small transformer proposes transformation primitives that effectively restrict the combinatorial search space;
- use synthetic data for both pre-training and test-time adaptation, and show that NSA improves over comparable DSL-based and hybrid baselines on the ARC train and evaluation sets.

2 Method

Our approach combines (i) a DSL for expressing ARC transformations, (ii) a small transformer that proposes likely transformation primitives from few-shot examples, and (iii) a symbolic search procedure that instantiates and composes these primitives.

2.1 DSL & Symbolic Engine

We build on the "Abstract Reasoning with Graph Abstractions" (ARGA) DSL [6], which represents grids as graphs and applies programs over these abstractions. The DSL has three components: (i) *abstract representations*, where connected pixel components become nodes and edges encode spatial relations; (ii) *filters*, which select nodes based on properties such as size, color, or neighborhood; and (iii) *transformation primitives*, which modify selected nodes (e.g., recoloring, rotating, extending). A full program is specified by choosing an abstraction and a sequence of filters with associated transformation primitives and parameters. ARGA provides 4 base filters and 12 primitives. To increase representational

ESANN 2026 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 22-24 April 2026, i6doc.com publ., ISBN 9782875870964. Available from http://www.i6doc.com/en/.

coverage we add 15 generic primitives, yielding the extended DSL *ARGAe*. The new primitives include grid-level operations (e.g. mirroring, rotation, upscaling), object operations (e.g. extract, duplicate, recolor, truncate), and layout operations (e.g. beam, connect, magnet).

Primitive	**Description**
extract	Extract a node
duplicate	Duplicate a grid along an axis
upscale_grid	Multiply grid size and expand pixels
fill	Fill a subgrid with an object
magnet	Move pixels to a border or node
beam	Draw a colored line in a direction
shift	Shift the grid in a direction
mirror_duplicate	Duplicate and mirror the grid
rotate_duplicate	Duplicate at several rotations
mirror_grid	Mirror along an axis
rotate_grid	Rotate the grid by an angle
connect	Connect objects with a colored line
recolor	Recolor an object
truncate	Remove object and restore background

Table 1: Additional transformation primitives in `ARGAe`. Full definitions are given in the supplementary material.

A richer DSL increases the number of ARC tasks that can be expressed, but also enlarges the search space. If used with unguided symbolic search, ARGAe can perform worse in practice than the original ARGA DSL because many more candidate programs must be explored.

Combinatorial Search For program execution we reuse ARGA's greedy best-first search [6]. Nodes in the search tree correspond to partially transformed grids; expansions apply one additional primitive (with parameters), and nodes are ordered by a distance measure to the target outputs. The search employs pruning constraints, state hashing, and a tabu list to avoid redundant or unpromising programs. In our neuro-symbolic variant, the set of candidate primitives considered at each step is restricted using transformer predictions.

2.2 Transformer-Guided Search

To guide search, we train a transformer encoder to propose which transformation primitives appear in the (unknown) program solving a task. Given the few input–output pairs of a single ARC task, the model predicts (i) how many primitives are used (up to three) and (ii) which primitives are applied at each step.
We use a standard Transformer encode with a custom grid tokenizer. The grid is serialized into a sequence of tokens encoding colors and structural markers (new row, new input, new output). We append three classification tokens, one

ESANN 2026 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 22-24 April 2026, i6doc.com publ., ISBN 9782875870964. Available from http://www.i6doc.com/en/.

per potential transformation. Each classification head outputs a distribution over the ARGAe primitives plus a special `no_trans` symbol indicating that no further primitive is required. In practice, three steps suffice to cover almost all ARGAe-solvable tasks we encountered. At inference time, the top-k primitives for each step define a small candidate set that constrains the symbolic search. This retains the representational power of ARGAe while greatly reducing the effective branching factor.

2.3 Synthetic Training and Test-Time Adaptation

ARC provides only 400 training tasks, which is insufficient to train the proposal model. We therefore generate large synthetic datasets by applying random ARGAe programs to real grids.

Synthetic Task Generation To create a synthetic task, we sample an ARC task, keep only its input grids, and draw a random sequence of abstractions, filters, primitives, and parameters from ARGAe. Applying this program to all inputs yields corresponding outputs and a labeled synthetic task. Programs that leave every input unchanged are discarded, and duplicate tasks are removed.

Pre-Training and Test-Time Adaptation We pre-train the transformer on 31 125 synthetic tasks using a cross-entropy loss over primitive predictions. At test time, we adapt the model to each ARC task by generating synthetic tasks from that task's input grids and fine-tuning on this task-specific synthetic set. The adapted transformer then produces proposals for the symbolic search, all within the same overall compute budget as prior ARC systems.

3 Experiments

We evaluate our neuro-symbolic approach on the ARC train and evaluation sets [2]. We show that (i) extending the DSL alone hurts pure symbolic search, (ii) transformer-guided proposals recover and exceed the original ARGA performance, and (iii) test-time adaptation (TTA) further improves results.

Data Generation and Training Setup We pre-train the proposal transformer on 31 125 synthetic tasks generated by hindsight relabeling (Section 2): input grids are sampled from both ARC train and eval sets, and random ARGAe programs are applied to produce labeled input–output pairs, without ever using ground-truth ARC solutions. Roughly 40% of the synthetic tasks use one primitive, 40% two, and 20% three. The transformer encoder has 8 layers, hidden size 512, feedforward size 2048, 8 heads, and about 25.3M parameters. We train with Adam at learning rate $5 \cdot 10^{-5}$ and batch size 32. For TTA we fine-tune the same model for 15 epochs on 2 500 synthetic tasks generated from the current ARC task's inputs.

Our Variants We report three variants:

`ARGAe`**:** Pure symbolic search using the extended DSL with all filters and primitives and the original ARGA search engine [6].

ESANN 2026 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 22-24 April 2026, i6doc.com publ., ISBN 9782875870964. Available from http://www.i6doc.com/en/.

Method	Approach	ARC Train Set	ARC Eval Set
Ferre [4]	DSL	29/400	6/400
Ferre [3]	DSL	96/400	23/400
Ainooson (MLE) [8]	DSL	70/400	17/400
Ainooson (Brute Force) [8]	DSL	104/400	26/400
CodeIt [1]	DSL & ML	-	59/400
ARGA [6]	DSL	50/400	9/400
`ARGAe`	DSL	6/400	22/400
`NSA w/o TTA`	DSL & ML	48/400	63/400
`NSA` (ours)	DSL & ML	78/400	75/400
Mirchandani [9]	ML	56/400	27/400
Greenblatt [7]	ML	-	168/400
BARC [5]	ML	-	227/400

Table 2: Solved ARC tasks on the train and evaluation sets. Best results are colored red, second best orange, third best yellow.

`NSA w/o TTA`: Transformer-guided search using the pre-trained proposal model only. We predict up to three primitives from ARGAe and restrict the search to those.

`NSA`: Full neuro-symbolic system with TTA. For each task, we fine-tune on synthetic data generated from the task's inputs and then run the same transformer-guided search as in `NSA w/o TTA`.

Baselines We compare to recent DSL-based systems [4, 3, 8, 6] and to the DSL+LLM approach CodeIt [1]. For context we also report results from purely ML-based ARC systems using large language models [9, 7, 5], although these typically require substantially more compute or sampling than our small transformer. All DSL-based methods rely on an explicit program search over a DSL; our variants differ only in the DSL (ARGA vs. ARGAe) and whether search is guided by the transformer and TTA.

Evaluation Protocol For each ARC task we allow up to 30 minutes of wall-clock time, as in [6]. This budget includes synthetic data generation, TTA fine-tuning, and symbolic search. In contrast to methods that try multiple candidate programs and only count the best one, we make a single final prediction per task, similar to [9]. We report the number of tasks solved (all test outputs correct) on both the train and evaluation sets.

Results and Discussion Table 2 shows that `NSA` substantially improves over comparable DSL and DSL+ML approaches that operate under similar resource constraints. On the eval set, `NSA` solves 75 tasks, compared to 59 for CodeIt and 26 for the best purely symbolic baseline [8]. On the train set, `NSA` reaches 78 solved tasks, placing third among DSL-based methods but with similar performance on train and eval, unlike many baselines that drop sharply on the evaluation set. The behavior of `ARGAe` highlights the trade-off between DSL ex-

ESANN 2026 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 22-24 April 2026, i6doc.com publ., ISBN 9782875870964. Available from http://www.i6doc.com/en/.

pressiveness and search complexity: despite its higher representational capacity, `ARGAe` solves far fewer tasks than ARGA on the train set, as the enlarged search space overwhelms unguided search. Transformer-guided search in `NSA w/o TTA` largely recovers this loss and already outperforms prior DSL methods on the eval set (63 tasks). Adding TTA in `NSA` further boosts performance by refining proposals to each task.

4 Conclusion

We introduced a neuro-symbolic approach for the ARC challenge, showing that combining the abstraction capabilities of hand-designed DSLs with a learned proposal model outperforms comparable purely hand-designed search methods. A straightforward extension is to scale up both the DSL and the transformer, for example using pre-trained LLMs and more training data, but this would soon exceed the 30-minute compute budget. Another direction is to iteratively refine candidate transformations by analysing their errors, analogous to iterative LLM-based approaches.

References

[1] Butt, Natasha and Manczak, Blazej and Wiggers, Auke and Rainone, Corrado and Zhang, David and Defferrard, Michaël and Cohen, Taco. CodeIt: Self-Improving Language Models with Prioritized Hindsight Replay. In *International Conference on Machine Learning*, PMLR, 2024. arXiv:2402.04858. Available at `https://arxiv.org/abs/2402.04858`.

[2] François Chollet. On the Measure of Intelligence. arXiv preprint arXiv:1911.01547, 2019. Available at `https://arxiv.org/abs/1911.01547`.

[3] Ferré, Sébastien. Tackling the abstraction and reasoning corpus (ARC) with object-centric models and the MDL principle. In *International Symposium on Intelligent Data Analysis*, pages 3–15. Springer, 2024.

[4] Ferré, Sébastien. First steps of an approach to the ARC challenge based on descriptive grid models and the minimum description length principle. arXiv preprint arXiv:2112.00848, 2021.

[5] Li, Wen-Ding and Hu, Keya and Larsen, Carter and Wu, Yuqing and Alford, Simon and Woo, Caleb and Dunn, Spencer M and Tang, Hao and Naim, Michelangelo and Nguyen, Dat and others. Combining induction and transduction for abstract reasoning. arXiv preprint arXiv:2411.02272, 2024.

[6] Xu, Yudong and Khalil, Elias B and Sanner, Scott. Graphs, constraints, and search for the abstraction and reasoning corpus. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, number 4, pages 4115–4122, 2023.

[7] Greenblatt, Ryan. Getting 50% (SoTA) on ARC-AGI with GPT-4o. 2024. Available at `https://redwoodresearch.substack.com/p/getting-50-sota-on-arc-agi-with-gpt`.

[8] Ainooson, James and Sanyal, Deepayan and Michelson, Joel P and Yang, Yuan and Kunda, Maithilee. An approach for solving tasks on the Abstract Reasoning Corpus. arXiv preprint arXiv:2302.09425, 2023.

[9] Suvir Mirchandani and Fei Xia and Pete Florence and Brian Ichter and Danny Driess and Montserrat Gonzalez Arenas and Kanishka Rao and Dorsa Sadigh and Andy Zeng. Large Language Models as General Pattern Machines. arXiv preprint arXiv:2307.04721, 2023. Available at `https://arxiv.org/abs/2307.04721`.