Neuro-Symbolic Mapping: Translating Visual Relational Triggers into Discrete DSL Programs

Ayan Tuteja[1]

Received June 1, 2026
Accepted September 2, 2026
Electronic access September 30, 2026

Background/Objective: This study examines a controlled visual-to-program prediction task inspired by ARC-style relational reasoning. The task contains six valid four-token programs and is intended as a narrow testbed for studying sequence prediction, positional encoding, and shortcut sensitivity rather than as a direct evaluation on ARC-AGI.
Methods: We train an autoregressive Transformer Executor to map flattened 15×15 color grids to programs in a restricted domain-specific language (DSL). The model is compared with a two-layer convolutional neural network (CNN) using global average pooling on a standard holdout set. We also evaluate the Transformer on a separately generated counterfactual set in which sampled object count conflicts with pixel mass. Both models are evaluated on the same counterfactual set to test sensitivity to the pixel-mass shortcut.
Results: In one reported training run, the Transformer achieved 100.0% Exact Sequence Match (ESM) on the 1,000-example standard holdout set, compared with 86.6% for the CNN baseline. On a separately generated 1,000-example counterfactual set, the Transformer achieved 984/1,000 correct sequences (98.4% ESM), whereas the CNN baseline declined to 125/1,000 correct sequences (12.5% ESM). These single-run results do not include variance estimates.
Conclusions: The results show that the Transformer can fit this restricted six-template visual-to-DSL task and remains accurate on the counterfactual set, whereas the CNN global-pooling baseline collapses when pixel mass conflicts with object count. The evidence is preliminary because only one training run is reported, the counterfactual generator is unseeded and class-imbalanced, and no stronger spatial baseline is evaluated.

Keywords: visual-to-DSL prediction, synthetic reasoning task, Transformer, CNN baseline, shortcut sensitivity, program generation.

Introduction

Background and Context

ARC-AGI motivates research on generalization and compositional reasoning from sparse examples[1]. The present work does not evaluate full ARC tasks. Instead, it studies a closed-world synthetic task with six valid DSL programs, providing a controlled setting for examining visual-to-program prediction and potential shortcut sensitivity.

Problem Statement and Rationale

Neural models can exploit correlations that are predictive in training data without capturing the intended decision rule. Torralba and Efros described dataset bias[2], and Geirhos et al. formalized shortcut learning as a broader concern for distribution shift[3]. The controlled task studied here uses a pixel-mass conflict as one diagnostic, but it does not establish general symbol grounding or ARC-level reasoning.

Significance and Purpose

We present an exploratory comparison of a Transformer Executor and a CNN global-pooling baseline on a restricted visual-to-DSL task. The contributions are limited to the following: Architecture Documentation: We specify a Transformer Executor with three encoder layers, three decoder layers, standard decoder cross-attention, PyTorch post-norm LayerNorm, one-dimensional sinusoidal positional encoding over the flattened 225-token grid sequence, and causal masking. Restricted Counterfactual Evaluation: We report Transformer performance on a generated counterfactual set in which sampled object count conflicts with pixel mass. Both the Transformer and the CNN baseline are evaluated on the same counterfactual examples. Training-Scale Observation: In one reported run, the 5,000-example condition converged faster and reached higher standard-holdout ESM than the 1,000-example

[1] *Stratford Preparatory Blackford, San Jose, CA, USA*

condition. This observation is not a statistically validated data-sufficiency threshold.

Objectives and Research Questions

This exploratory study asks four questions: (1) Can the Transformer learn the six valid four-token DSL templates in the reported standard split? (2) How does performance differ between the 1,000- and 5,000-example conditions in the reported run? (3) Does the Transformer remain accurate on the generated pixel-mass-conflict set? (4) Does the CNN global-pooling baseline degrade on the same counterfactual examples?

Scope and Limitations

The empirical claims are limited by the closed-world six-template DSL, deterministic object-processing rules, one reported training run, an unseeded and imbalanced counterfactual generator, and the absence of a stronger spatial baseline. Expressivity Constraints of the DSL: The domain-specific language used in this study is a closed-world system restricted to linear sequences with two decision tiers. It does not include iterative loops, dynamically created subroutines, or unconstrained conditional branching. Adding such operators would substantially increase the search space and optimization difficulty.

Theoretical Framework

The task is motivated by work on structured representations, object-centric processing, and neuro-symbolic reasoning[4–9]. However, the present experiment does not demonstrate human-like core knowledge or general symbol grounding[10]. It evaluates whether a neural sequence model can predict one of six fixed program structures from synthetic color grids processed under deterministic rules. Prior work on neuro-symbolic concept learning[6], program induction[11], program sketches[12] and neural sequence-to-sequence models for DSLs[13,14] motivates structured prediction over executable programs evaluated on synthetic visual diagnostic datasets[15,16]. In this study, the Transformer is compared only with a CNN that applies global average pooling. That baseline removes detailed spatial layout before classification, so the comparison should not be interpreted as evidence that self-attention is necessary or superior to all convolutional or vision architectures. Stronger baselines that preserve spatial information are required. The 15×15 grid is flattened into 225 color tokens. Each token is mapped to a 256-dimensional embedding, and a one-dimensional sinusoidal positional encoding is added according to flattened sequence position. A three-layer Transformer encoder processes the sequence, and a three-layer autoregressive decoder with standard cross-attention predicts a fixed four-token DSL program. The experiment is therefore closer to constrained visual sequence classification than to open-ended program synthesis.

Methodology Overview

We report descriptive ESM results for the Transformer and CNN on the standard holdout set and for the Transformer on the counterfactual set. Because only one training run is available, no means, standard deviations, confidence intervals, or significance tests are reported. Both models are evaluated on the same counterfactual examples. The experiment is inspired by ARC-style relational reasoning but does not use the ARC format of inferring unseen transformations from multiple input-output demonstrations. It is a six-class, fixed-length visual-to-DSL benchmark.

Methods

Research Design

The generated DSL sequence has a fixed length of $L = 4$ and follows the grammar Program $\rightarrow \mathrm{Op}_1 \rightarrow \mathrm{Op}_2 \rightarrow \mathrm{Op}_3 \rightarrow \mathrm{Op}_4$. The DSL contains seven operators: partition, filter-color-red, filter-color-blue, sort-size-desc, get-first, get-last, and to-coords. Each valid program begins with partition, applies one of three second-step relational operations (filter-color-red, filter-color-blue, or sort-size-desc), selects either the first or last object, and ends with to-coords. This produces the six valid program templates listed in Table 1.

Synthetic Data

We procedurally generated datasets containing 1,000 and 5,000 examples. Each 15×15 grid becomes a sequence of 225 color tokens from a vocabulary of 10 values (0–9). Standard data use an 80/20 partition: the 1,000-example condition contains 800 training and 200 holdout examples, and the 5,000-example condition contains 4,000 training and 1,000 holdout examples. A separately generated counterfactual set contains 1,000 examples and is used to examine conflicts between object count and pixel mass. In one generated counterfactual set, the Transformer achieved 984/1,000 ESM, while the CNN baseline achieved 125/1,000. The available generator does not set random.seed or torch.manual_seed, so exact regeneration is not guaranteed. Object placements may overlap, no overlap-rejection rule is implemented, and class balance is not explicitly enforced.

Table 1 Six Valid DSL Program Templates

Template Class	Exact DSL Sequence	Operational Meaning
Red+First	`partition -> filter_color_red -> get_first -> to_coords`	Partition the grid, retain red objects, select the first object, and return its coordinates.
Red+Last	`partition -> filter_color_red -> get_last -> to_coords`	Partition the grid, retain red objects, select the last object, and return its coordinates.
Blue+First	`partition -> filter_color_blue -> get_first -> to_coords`	Partition the grid, retain blue objects, select the first object, and return its coordinates.
Blue+Last	`partition -> filter_color_blue -> get_last -> to_coords`	Partition the grid, retain blue objects, select the last object, and return its coordinates.
Sort+First	`partition -> sort_size_desc -> get_first -> to_coords`	Partition all objects, sort them by descending size, select the largest object, and return its coordinates.
Sort+Last	`partition -> sort_size_desc -> get_last -> to_coords`	Partition all objects, sort them by descending size, select the smallest object, and return its coordinates.

Data Collection

The counterfactual generator creates 0–3 red objects and 0–3 blue objects, for 0–6 total target objects. Red is color 1 and blue is color 2. When one color is the minority, its objects are drawn as 3×3 squares while the majority color uses 1×1 squares, creating a conflict between object count and pixel mass. When red and blue counts are equal, the Sort template is selected. The third DSL operator is get_first when the total sampled object count exceeds four and get_last otherwise. Objects are located using 8-way connectivity; diagonal adjacency therefore merges components. No minimum separation or rejection condition is imposed, overlaps are permitted and later slices may overwrite earlier ones, and irregular components can arise accidentally. Noise consists of 1–5 randomly placed pixels using colors 3–9; noise can connect and is not excluded from partition_grid. Equal-size ties are resolved by Python's stable descending sort using the top-to-bottom, left-to-right discovery order returned by scipy.ndimage.label. No class-balancing procedure, fixed generator seed, rejection sampling, or cue-label correlation audit is implemented. The counterfactual set contains 1,000 examples. Its class distribution is not controlled: independent random draws make get_last cases approximately 81% of the set and get_first cases approximately 19%. The Transformer achieved 984/1,000 correct sequences (98.4% ESM) and the CNN baseline achieved 125/1,000 correct sequences (12.5% ESM) on this same generated set. Because generation is unseeded and labels can disagree with the final rendered grid after overlap, merging, or noise connectivity, these numbers should be interpreted with appropriate caution.

Variables and Measurements

In autoregressive sequence generation, Exact Sequence Match (ESM) requires the resulting four-token program to match the target sequence exactly. During inference, the fixed initial partition operator is supplied to the decoder, and the model predicts the remaining three operators [17]. The principal evaluation and model quantities are defined below.

$$P(\text{ESM}) = \prod_{i=1}^{L} P(\hat{t}_i = t_i), \quad L = 4$$

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$\mathscr{L}_{CE} = -\sum_{x} p(x)\log q(x)$$

Here, $p(x)$ is the target distribution and $q(x)$ is the predicted distribution.

$$F_{\text{obj}}(G) = |CC_8(G)|$$

Here, $CC_8(G)$ denotes the set of 8-connected components in grid G.

Given the six valid program templates, uniformly random complete-template guessing yields an expected ESM of 1/6 (approximately 16.7%). Under the simplifying assumption that accuracies at the three predicted positions are equal and independent, 95% sequence-level ESM corresponds to approximately 98.3% accuracy at each predicted position. This calculation is illustrative; token errors need not be independent. High ESM indicates consistent sequence prediction within the tested distribution but does not establish the model's internal reasoning strategy.

Procedure

To evaluate the potential benefit of self-attention for global relational processing, we compared two neural architectures. The Transformer Executor [18] uses three encoder layers and three decoder layers with standard

nn.TransformerDecoderLayer cross-attention, a model and embedding dimension of 256, eight attention heads, PyTorch's default feedforward dimension of 2,048, ReLU activation, dropout of 0.1, and default post-norm LayerNorm. The input vocabulary contains 10 color values (0–9). Separate pixel_emb and token_emb modules map input colors and the eight DSL vocabulary entries; their weights are not shared. The 15 $\times$ 15 grid is flattened to 225 tokens, and a standard one-dimensional sinusoidal PositionalEncoding is added according to flattened sequence position. The output vocabulary contains eight TOKEN_MAP entries, including $\langle$PAD$\rangle$ = 0, and no $\langle$EOS$\rangle$ token. Greedy inference initializes the decoder sequence with token ID 1 (partition), which serves as both the initial decoder token and the first DSL operator, and appends three predicted operators to produce the fixed four-token program. A causal mask prevents attention to future output positions. The CNN baseline is nn.Embedding(10, 16) followed by Conv2d (16, 64, kernel_size=3, stride=1, padding=1), ReLU, Conv2d(64, 128, kernel_size=3, stride=1, padding=1), ReLU, AdaptiveAvgPool2d ((1, 1)), and a linear classifier. Both models were optimized with Adam using a learning rate of 0.0008 and a batch size of 64 [19].

Data Analysis

A critical methodological safeguard in the autoregressive formulation is prevention of forward-looking data leakage. During training, target shifting conditions the model on target[:, :-1] to predict target[:, 1:], and the Transformer decoder uses generate_square_subsequent_mask to block attention to future program tokens. Reported ESM values are computed with greedy inference rather than teacher-forced outputs. In greedy_inference, the sequence is initialized with token ID 1 (partition), which serves as the first DSL operator; the model then predicts the remaining three operators until the fixed four-token sequence is complete. The reported results include exact sequence match, loss, and class-level confusion matrices. Per-token precision, recall, and F1 are not reported and are therefore not inferred from the confusion matrices.

Ethical Considerations

As this study strictly utilizes procedurally generated synthetic arrays and logic grids, no human or animal participants were involved, and no specific ethical or institutional board approval was required.

Results

In one reported run, the Transformer reached 100.0% ESM by Epoch 2 on the 1,000-example standard holdout set, while the CNN global-pooling baseline achieved 86.6%. On one separately generated 1,000-example counterfactual set, the Transformer achieved 984/1,000 correct sequences (98.4% ESM), while the CNN baseline declined to 125/1,000 correct sequences (12.5% ESM). No variance across training seeds is reported.

CNN Confusion Matrix Analysis

The CNN exhibited substantial off-diagonal errors, particularly between the Red+First and Red+Last classes and between the Blue+First and Blue+Last classes, with 33/61 and 32/64 correct predictions in the two First-class subsets.

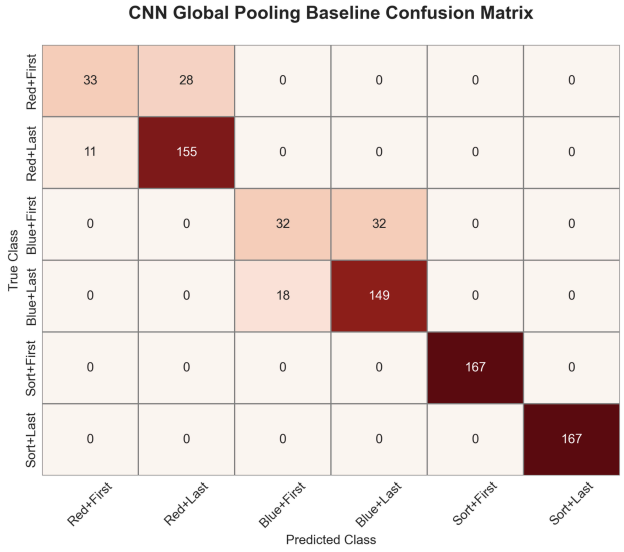

Figure. 1 CNN global-pooling baseline confusion matrix on the 1,000-example standard holdout set.

Transformer Confusion Matrix Analysis

The Transformer Executor produced a fully diagonal confusion matrix on the 1,000-example standard holdout set, correctly classifying all six program templates and yielding 100.0% standard ESM.

Discussion

Restatement of Key Findings

A central finding of this ablation study is the substantial improvement observed between the 1,000- and 5,000-sample conditions. In the reported run, the Transformer trained at the larger scale achieved 97.0% ESM in Epoch 1 and reached 100.0% by Epoch 2, with loss approaching zero. These results

Table 2 Descriptive Standard-Holdout and Counterfactual Results from One Reported Run

Model	Standard-Holdout ESM	Counterfactual ESM	Status
Transformer Executor	100.0% (1,000/1,000)	98.4% (984/1,000)	Verified from reported evaluation
CNN global-pooling baseline	86.6% (866/1,000)	12.5% (125/1,000)	Verified from reported evaluation

Table 3 Transformer Epoch Progression (5,000 Samples)

Epoch	Loss	ESM
Epoch 1	0.0548	97.0%
Epoch 2	0.0005	100.0%
Epoch 3	0.0002	100.0%
Epoch 4	0.0001	100.0%
Epoch 5	0.0001	100.0%

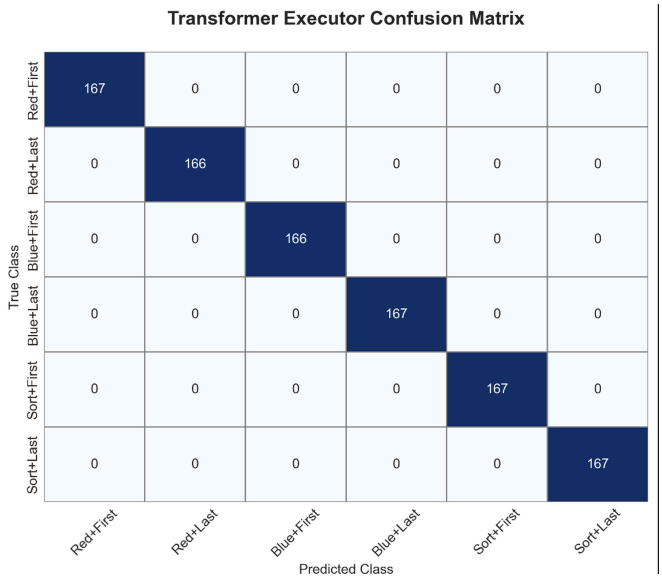

Figure. 2 Transformer Executor confusion matrix on the 1,000-example standard holdout set.

indicate that grammar acquisition in this controlled task is sensitive to training-data diversity. The larger synthetic dataset exposed the model to greater combinatorial variation, which may have reduced opportunities for rote memorization and encouraged alignment with the six program templates[6]. The architectural ablation also suggests that self-attention is useful for relational mapping in this setting. The CNN global-pooling baseline achieved 86.6% sequence match on the standard holdout set but showed pronounced confusion between the First and Last variants of the red and blue templates.

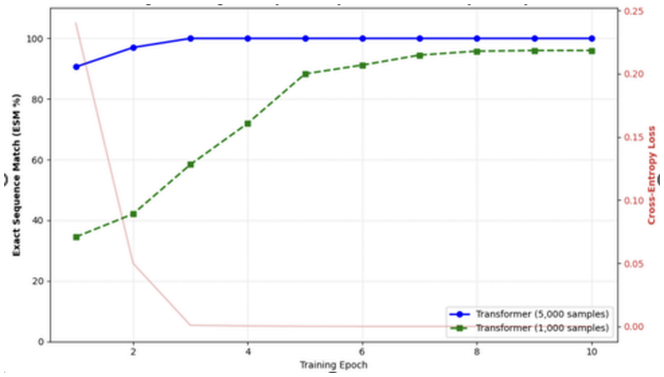

Figure. 3 Training progression in the reported run under the 1,000- and 5,000-sample conditions. The 5,000-sample model reached 97.0% ESM in Epoch 1 and 100.0% in Epoch 2, whereas the 1,000-sample model improved more gradually.

Implications and Significance

On the standard holdout set, the CNN global-pooling baseline showed confusion between the First and Last variants of the red and blue templates. This result characterizes only this specific architecture. Because global average pooling removes detailed spatial layout, the baseline is deliberately limited and should not support claims about CNNs generally or about the necessity of self-attention.

Counterfactual Performance

The Transformer achieved 984/1,000 correct sequences on one generated counterfactual set, indicating that the trained model often retained the intended template under the tested pixel-mass conflict. In contrast, the CNN global-pooling baseline declined to 125/1,000 correct sequences (12.5% ESM) on the same examples, frequently favoring the template of the larger-footprint color. Because global average pooling blends spatial detail into a single latent vector, this behavior is consistent with reliance on total pixel mass as a shortcut, echoing the dataset-bias and shortcut-learning concerns of Torralba and Efros[2] and Geirhos et al.[3], as well as related literature on spurious correlations and 'Clever Hans' predictors[20,21].

The evidence remains tempered by class imbalance, unseeded generation, overlapping objects, possible label/render mismatch, and noise participation in partitioning.

Restricted Program-Sequence Prediction

The resulting four-token output includes a fixed initial partition operator, a second-step choice among red filtering, blue filtering, and size sorting, a third-step choice between first and last, and the fixed final to_coords operator. During inference, partition is supplied and the model predicts the remaining three positions. Successful ESM therefore requires the complete resulting program sequence to match. Because only six valid programs exist, the task remains a small closed-world benchmark rather than evidence of broad compositional program synthesis.

Connection to Objectives

The reported results show that the Transformer learned the restricted six-template task in one run and performed strongly on one generated counterfactual set. They do not establish general symbol grounding, ARC-level reasoning, robustness across seeds, or superiority over stronger spatial baselines.

Recommendations and Future Work

The most important next step is to convert the current exploratory study into a reproducible benchmark by using fixed seeds, persisting datasets, deriving labels from final rendered grids, and balancing classes. Additional directions include:

Execution-Guided Verification (CEGIS/PoT): Currently, the model is evaluated via greedy decoding, which remains inherently vulnerable to cascading errors in autoregressive generation[17]. Future iterations will integrate Execution-Based Beam Search and stepwise verification, mirroring Program of Thoughts (PoT)[22], Counterexample-Guided Inductive Synthesis (CEGIS)[23], and REPL-based neural synthesis loops[24]. The model will generate top-K candidate programs and explicitly execute them against training examples; programs failing to produce the target output will be programmatically discarded, allowing closed-loop logical self-correction.

Bidirectional Inverse-Semantics Search: To reduce the massive combinatorial search space of program synthesis, bidirectional inverse-semantics search mechanisms are explored[25]. Implementing conditional-inverse operators within the DSL will allow the system to systematically reason top-down from the desired output grid back toward the input grid, encouraging true compositional generalization.

Few-Shot Meta-Learning and MAML Integration: To align with the sparse, few-shot induction requirements of the original ARC benchmark[1], we must restructure the input pipeline for dynamic context windows. Following Chelsea Finn's principles of Model-Agnostic Meta-Learning (MAML)[26], shifting toward in-context learning may encourage the model to infer rules dynamically from limited visual examples and support more realistic few-shot reasoning. Curriculum-based task progression may also be evaluated[27].

Additional extensions include loop-capable DSL operators and self-supervised segmentation modules. Their value should be assessed empirically because both changes would expand the search space and alter the model's object representation.

Limitations

Task and Framing Limitations: The experiment contains only six valid fixed-length programs, and a uniform complete-template baseline achieves approximately 16.7%. The task is therefore substantially narrower than ARC-AGI and related benchmarks such as ConceptARC[28], and should be interpreted as a controlled visual-to-DSL benchmark rather than a general reasoning or grounding result.

Rendered-Grid Label Consistency: Targets are selected from sampled object counts before rendering, while overlaps, overwriting, diagonal merging, and noise connectivity can alter the final connected components visible in the grid. Consequently, some labels may not match the final rendered input. Future datasets should compute connected components after rendering and derive or validate the target program from that final grid.

Single-Run and Baseline Limitations: All reported training results come from one run, so stability across initialization and sampling is unknown. The only baseline evaluated uses global average pooling. Future work should report multiple independent seeds and compare against stronger baselines such as a CNN that preserves spatial features, a coordinate-aware CNN, or a small vision Transformer.

Hardcoded Object-Centric Priors: Finally, the system relies entirely on a hardcoded 8-way connectivity algorithm to establish its fundamental object-centric priors[5]. While mathematically mirroring cognitive theories of objectness, it bypasses the challenge of having the neural network autonomously learn what constitutes an object in noisy environments. Future iterations would benefit from integrating learned segmentation backbones or unsupervised object-centric architectures[29,30] to dynamically construct explicit state graphs based on localized contexts.

Closing Thought

This exploratory study evaluated a restricted six-template visual-to-DSL task. In one reported run, the Transformer achieved 100.0% ESM on the 1,000-example standard holdout set and 984/1,000 correct sequences (98.4% ESM) on one generated counterfactual set; the CNN global-pooling baseline achieved 86.6% on the standard holdout set but declined to 125/1,000 correct sequences (12.5% ESM) on the same counterfactual set. These results indicate that, under the tested con-

ditions, the Transformer was substantially less dependent on the pixel-mass shortcut. These observations demonstrate task-specific performance, not general symbol grounding or ARC-level reasoning.

References

1 F. Chollet, On the measure of intelligence. arXiv preprint arXiv:1911.01547, 2019.

2 A. Torralba and A. A. Efros, Unbiased look at dataset bias. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pg. 1521–1528, 2011., https://doi.org/10.1109/CVPR.2011.5995347.

3 R. Geirhos, J. H. Jacobsen, C. Michaelis, R. Zemel, W. Brendel, M. Bethge, and F. A. Wichmann, Shortcut learning in deep neural networks. Nature Machine Intelligence. Vol. 2, pg. 665–673, 2020., https://doi.org/10.1038/s42256-020-00257-z.

4 P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, C. Gulcehre, F. Song, A. Ballard, J. Gilmer, G. Dahl, A. Vaswani, K. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu, Relational inductive biases, deep learning, and graph networks. arXiv preprint arXiv:1806.01261, 2018.

5 E. S. Spelke, Principles of object perception. Cognitive Science. Vol. 14, pg. 29–56, 1990., https://doi.org/10.1207/s15516709cog1401_3.

6 J. Mao, C. Gan, P. Kohli, J. B. Tenenbaum, and J. Wu, The neuro-symbolic concept learner: interpreting scenes, words, and sentences from natural supervision. International Conference on Learning Representations, 2019.

7 K. Greff, S. van Steenkiste, and J. Schmidhuber, On the binding problem in artificial neural networks. arXiv preprint arXiv:2012.05208, 2020.

8 J. B. Tenenbaum, C. Kemp, T. L. Griffiths, and N. D. Goodman, How to grow a mind: statistics, structure, and abstraction. Science. Vol. 331, pg. 1279–1285, 2011., https://doi.org/10.1126/science.1192788.

9 M. Garnelo and M. Shanahan, Reconciling deep learning with symbolic artificial intelligence. Current Opinion in Behavioral Sciences. Vol. 29, pg. 17–21, 2019., https://doi.org/10.1016/j.cobeha.2019.02.002.

10 S. Harnad, The symbol grounding problem. Physica D: Nonlinear Phenomena. Vol. 42, pg. 335–346, 1990., https://doi.org/10.1016/0167-2789(90)90087-6.

11 K. Ellis, C. Wong, M. Nye, M. Sablé-Meyer, L. Morales, L. Hewitt, L. Cary, A. Solar-Lezama, and J. B. Tenenbaum, DreamCoder: bootstrapping inductive program synthesis with wake-sleep library learning. PLOS Computational Biology. Vol. 17, pg. e1008926, 2021., https://doi.org/10.1371/journal.pcbi.1008926.

12 M. Nye, L. Hewitt, J. B. Tenenbaum, and A. Solar-Lezama, Learning to infer program sketches. Proceedings of the 36th International Conference on Machine Learning. Vol. 97, pg. 4861–4870, 2019.

13 J. Devlin, J. Uesato, S. Bhupatiraju, R. Singh, A. Mohamed, and P. Kohli, RobustFill: Neural program learning under noisy I/O. Proceedings of the 34th International Conference on Machine Learning. Vol. 70, pg. 990–998, 2017.

14 M. Balog, A. L. Gaunt, M. Brockschmidt, S. Nowozin, and D. Tarlow, DeepCoder: Learning to write programs. International Conference on Learning Representations, 2017.

15 J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. L. Zitnick, and R. Girshick, CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pg. 2901–2910, 2017.

16 K. Yi, J. Wu, C. Gan, A. Torralba, P. Kohli, and J. B. Tenenbaum, Neural-symbolic VQA: Disentangling reasoning from vision and language understanding. Advances in Neural Information Processing Systems. Vol. 31, 2018.

17 S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, Scheduled sampling for sequence prediction with recurrent neural networks. Advances in Neural Information Processing Systems. Vol. 28, 2015.

18 A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, Attention is all you need. Advances in Neural Information Processing Systems. Vol. 30, pg. 5998–6008, 2017.

19 D. P. Kingma and J. Ba, Adam: a method for stochastic optimization. International Conference on Learning Representations, 2015.

20 S. Lapuschkin, S. Wäldchen, A. Binder, G. Montavon, W. Samek, and K.-R. Müller, Unmasking Clever Hans predictors and assessing what machines really learn. Nature Communications. Vol. 10, pg. 1096, 2019., https://doi.org/10.1038/s41467-019-08987-4.

21 M. Arjovsky, L. Bottou, I. Gulrajani, and D. Lopez-Paz, Invariant risk minimization. arXiv preprint arXiv:1907.02893, 2019.

22 W. Chen, X. Ma, X. Wang, and W. W. Cohen, Program of thoughts prompting: disentangling computation from reasoning. arXiv preprint arXiv:2211.12588, 2022.

23 A. Solar-Lezama, Program synthesis by sketching. Ph.D. dissertation, University of California, Berkeley, 2008.

24 K. Ellis, M. Nye, Y. Pu, F. Sosa, J. Tenenbaum, and A. Solar-Lezama, Write, execute, assess: Program synthesis with a REPL. Advances in Neural Information Processing Systems. Vol. 32, 2019.

25 S. Alford, A. Gandhi, A. Rangamani, A. Banburski, T. Wang, S. Dandekar, J. Chin, T. Poggio, and P. Chin, Neural-guided, bidirectional program search for abstraction and reasoning. arXiv preprint arXiv:2110.11536, 2021.

26 C. Finn, P. Abbeel, and S. Levine, Model-agnostic meta-learning for fast adaptation of deep networks. Proceedings of the 34th International Conference on Machine Learning. Vol. 70, pg. 1126–1135, 2017.

27 Y. Bengio, J. Louradour, R. Collobert, and J. Weston, Curriculum learning. Proceedings of the 26th International Conference on Machine Learning. pg. 41–48, 2009., https://doi.org/10.1145/1553374.1553380.

28 A. Moskvichev, V. V. Odouard, and M. Mitchell, The ConceptARC benchmark: evaluating understanding and generalization in the ARC domain. Transactions on Machine Learning Research, 2023.

29 F. Locatello, D. Weissenborn, T. Unterthiner, A. Mahendran, G. Heigold, S. Uszkoreit, A. Dosovitskiy, and T. Kipf, Object-centric learning with slot attention. Advances in Neural Information Processing Systems. Vol. 33, pg. 11527–11538, 2020.

30 C. P. Burgess, L. Matthey, N. Watters, R. Kabra, I. Higgins, M. Botvinick, and A. Lerchner, MONet: Unsupervised scene decomposition and representation. arXiv preprint arXiv:1901.11390, 2019.