A Neural Affinity Framework for Abstract Reasoning: Diagnosing the Compositional Gap in Transformer Architectures via Procedural Task Taxonomy

Miguel Ingram Arthur Merritt

November 2025

Keywords: Abstract Reasoning, ARC Benchmark, Compositional Generalization, Task Taxonomy, Neural Affinity, Transformer Limitations, Procedural Task Generation

Abstract

Responding to Hodel et al.'s (2024) call for a formal definition of task relatedness in `re-arc`, we present the first **9-category taxonomy** of all 400 tasks, validated at 97.5% accuracy via rule-based code analysis. We prove the taxonomy's visual coherence by training a CNN on raw grid pixels (**95.24% accuracy on S3, 36.25% overall, 3.3× chance**), then apply the taxonomy diagnostically to the original ARC-AGI-2 test set. Our curriculum analysis reveals **35.3%** of tasks exhibit low **neural affinity** for Transformers—a distributional bias mirroring ARC-AGI-2. To probe this misalignment, we fine-tuned a 1.7M-parameter Transformer across 186 tasks, revealing a profound **Compositional Gap**: **210 of 302 tasks (69.5%) achieve >80% cell accuracy (local patterns) but <10% grid accuracy (global synthesis)**. This provides direct evidence for a **Neural Affinity Ceiling Effect**, where performance is bounded by architectural suitability, not curriculum. **Applying our framework to Li et al.'s independent ViTARC study (400 specialists, 1M examples each) confirms its predictive power: Very Low affinity tasks achieve 51.9% versus 77.7% for High affinity (p<0.001), with a task at 0% despite massive data.** The taxonomy enables precise diagnosis: low-affinity tasks (A2) hit hard ceilings, while high-affinity tasks (C1) reach 99.8%. These findings indicate that progress requires **hybrid architectures** with affinity-aligned modules. **We release our validated taxonomy, classifiers, and reproduction package.**

M. Ingram: Corresponding author. Email: miguel.ingram.research@gmail.com. Independent researcher. Conducted all experimental analysis and wrote the manuscript.
A. Merritt: Independent researcher. Designed the champion model architecture.

Available at `https://github.com/ImmortalDemonGod/arc-taxonomy-reproduction`

1. Introduction

The Abstraction and Reasoning Corpus (ARC) (Chollet 2019) presents a formidable and enduring challenge to modern artificial intelligence. It was designed not merely to test performance but to measure a more general, human-like form of fluid intelligence that transcends mere pattern recognition—a goal that remains largely unmet. The persistent chasm between human intuition and machine performance on ARC highlights a fundamental misalignment between the dominant Transformer architecture and the demands of abstract reasoning. This suggests that progress is not merely a matter of scaling but requires a deeper, more principled approach to architectural design. This paper introduces a systematic, empirical framework for diagnosing this misalignment and charting a path toward more capable reasoning systems.

While state-of-the-art methods have made progress, they often function as complex engineering solutions, leaving a critical **diagnostic gap**: the community lacks a systematic methodology to understand *why* models fail on some tasks and succeed on others. Without a formal vocabulary for task types and their corresponding architectural requirements, research risks devolving into a cycle of brute-force scaling and ad-hoc engineering. Our work fills this gap by providing a comprehensive diagnostic toolkit, enabling researchers to move from simply measuring performance to understanding the root causes of architectural failure and success.

To fill this diagnostic gap in a principled way, our methodology directly implements the theoretical framework for measuring intelligence articulated in Chollet's "On the Measure of Intelligence" (Chollet 2019). This framework posits that true intelligence is not skill at known tasks but rather the *skill-acquisition efficiency* of a system when faced with novelty, an efficiency that is deeply dependent on its *priors* (innate knowledge) and the *generalization difficulty* of the tasks. Our approach operationalizes each of these concepts to create a fair and insightful measure of architectural capability. Specifically, we use large-scale pre-training on `re-arc` not to "buy skill" on the test set, but to **forge the Core Knowledge priors** (e.g., objectness, topology, geometry) that ARC assumes but which a "blank slate" Transformer lacks. With this prior-endowed model as a fair baseline, our fine-tuning experiments then directly measure **skill-acquisition efficiency** on novel tasks. Finally, our taxonomy serves to **deconstruct generalization difficulty** into a spectrum of measurable computational primitives. This alignment transforms what might appear as "teaching the test" into a rigorous, philosophically-grounded measurement of architectural affinity.

Built on this foundation, our primary contribution is a two-part diagnostic framework. First, we developed the first systematic **9-category taxonomy** of all 400 `re-arc` tasks (Hodel 2024), providing a formal language for "task relatedness." A critical and valid concern with any taxonomy derived from programmatic generator code is that it may not generalize to the visual domain of ARC itself. We address this head-on with a rigorous two-stage validation. The taxonomy is first validated with **97.5% accuracy** by a rule-based classifier on the generator code. We then prove its relevance to the visual domain by training a separate CNN-based classifier on raw `re-arc` grid pairs. This purely visual classifier successfully learns to distinguish our categories from pixel data alone—achieving **95.24% accuracy on the dominant S3 (Topological) category**—and can subsequently apply these labels to tasks from the **original ARC-AGI-2 benchmark**, confirming our framework's applicability to human-designed problems.

With this validated "map" of the ARC problem space, we demonstrate its diagnostic power. Our first application is to the `re-arc` curriculum itself, revealing a significant structural bias: **35.3%** of tasks fall into categories with low documented neural affinity for standard Transformers. Our second, more crucial application is to diagnose the performance of our 1.7M parameter champion model. This analysis uncovers the central empirical finding of this paper: a profound **"Compositional Gap."** We define this as the stark dissociation between a model's ability to learn local patterns (measured by cell-level accuracy) and its inability to compose them into a globally correct solution (grid-level accuracy). The evidence for this gap is overwhelming: after extensive, task-specific fine-tuning on **302 tasks**, **210 tasks (69.5%)** exhibit high local understanding (>80% cell accuracy) while simultaneously failing to achieve global correctness (<10% grid accuracy). This pattern, where the model masters the parts but fails the whole, provides a mechanistic explanation for the performance ceilings observed in ARC.

Ultimately, the value of any framework developed on `re-arc` must be measured by its ability to explain performance on the original, human-designed ARC benchmark. Therefore, our analysis culminates by applying our best pre-trained model to the **ARC-AGI-2 test set**. We confront the synthetic-to-real **generalization gap** directly, where grid accuracy drops precipitously from **2.34% on `re-arc` to 0.34% (95% CI: [0.18%, 0.49%]) on ARC-AGI-2**. Crucially, we then deploy our taxonomy as a diagnostic tool to explain this failure. Our analysis of the ARC-AGI-2 test set reveals that the most visually structured tasks—those for which our classifier is most confident—are **100% concentrated in the S2 and S3 categories**. Given that these categories are known from our `re-arc` analysis to be challenging for compositional synthesis, this strongly suggests that a significant portion of ARC-AGI-2's core difficulty lies in these low-to-medium affinity domains, validating our taxonomy's predictive power on real-world generalization failures.

To test whether our Neural Affinity Framework generalizes beyond our own model, we **apply it as a predictive lens to ViTARC** (Li et al. 2024), an independent study that trained 400 specialist Vision Transformers (one per task) with up to 1 million examples each. Despite architectural enhancements (2D-aware representations, object-based positional encodings) and massive data, our framework successfully predicts the observed performance patterns: **Very Low affinity tasks (A1, A2) achieve only 51.9% mean solve rate versus 77.7% for High affinity tasks (C1)**—a statistically significant 25.8 percentage point gap (p < 0.001, Cohen's d = 0.726). Critically, their results include a "smoking gun" task (137eaa0f, A2: Spatial Packing) that achieves **0.0% success despite 1 million training examples**, providing definitive evidence that the ceiling is architectural, not data-limited.

Core Thesis

> *The persistent failure of standard Transformers on ARC is not a matter of scale, data, or curriculum interference, but a fundamental architectural misalignment. By systematically classifying the ARC curriculum—and validating that our classification applies to the visual tasks themselves—we empirically prove that a* ***"Compositional Gap"*** *is the primary failure mode. This gap, which explains the* ***"Neural Affinity Ceiling Effect,"*** *persists from the synthetic* `re-arc` *domain to the real ARC-AGI-2 benchmark, where our taxonomy serves as a powerful diagnostic tool for predicting and explaining generalization failures.*

This diagnostic framework also contextualizes recent advances in the field. For instance, reported gains from architectures incorporating recurrence (e.g., TRM Jolicoeur-Martineau (2025)) can be interpreted through our lens as affinity-aligned modules targeted at iterative (A1) deficits. Our framework thus offers a principled hypothesis for why such designs help—recurrence supplies the learned iteration that fixed-depth Transformers lack—while we defer a full, systematic validation to future work.

This paper makes the following contributions:

a. **A Systematic and Dually-Validated Taxonomy:** We introduce the first 9-category taxonomy for `re-arc`, validated at both the code-level (97.5% accuracy) and the visual-level (95.24% on S3, 36.25% overall 9-way), and demonstrate its applicability to the original ARC-AGI-2 benchmark.

b. **Quantitative Evidence for Curriculum Bias:** We provide the first systematic analysis of the `re-arc` curriculum, revealing that 35.3% of tasks fall into low neural affinity categories for standard Transformers.

c. **Empirical Evidence for the Compositional Gap:** We provide the first quantitative evidence for the "Compositional Gap" as the primary failure mode for standard Transformers on ARC, where 210 of our 302 fine-tuned tasks (69.5%) achieve >80% cell-level accuracy but <10% grid-level accuracy.

d. **A Diagnostic Framework for Architectural Ceilings:** We introduce the Neural Affinity Ceiling Effect as a mechanism to explain these failures and demonstrate its predictive power on the ARC-AGI-2 generalization gap.

e. **Predictive Power on Independent Data:** We demonstrate our framework successfully predicts performance patterns in Li et al.'s independent ViTARC study (400 specialist models, 1M examples each), where Very Low affinity tasks achieve 25.8 percentage points lower performance than High affinity tasks ($p < 0.001$), including a task at 0% despite massive data.

f. **Evidence for Transfer Despite the Gap:** We demonstrate that pre-training on `re-arc` improves ARC-AGI-2 surrogate metrics (cell accuracy: 71.6% → 89.5%), proving that priors transfer even when compositional synthesis fails.

g. **A Publicly Released Toolkit:** We release our validated taxonomy, classifiers, datasets, and reproduction package to accelerate community research on architectural diagnostics.

A Critical Methodological Note on Scope: Our taxonomy is fundamentally a classification of the groups of computational primitives implemented in the `re-arc` procedural generators—a systematic reverse-engineering of Hodel et al.'s design choices. While our dual validation (code + visual) and external transfer experiments demonstrate that these categories capture meaningful patterns that generalize to human-designed ARC tasks, the taxonomy is contingent on the specific implementation of `re-arc`. A different procedural generation effort might yield different categorical boundaries. We therefore position this work as providing a validated taxonomy of the `re-arc` distribution, with strong empirical evidence (via visual classifier transfer to ARC-AGI-2 and external validation on ViTARC) that it captures broadly relevant reasoning primitives, rather than claiming it is a definitive, universal decomposition of abstract reasoning itself.

The remainder of this paper is structured as follows: Section 2 establishes the theoretical foundations of neural affinity and architectural limitations; Section 3 provides background on ARC and `re-arc`; Section 4 reviews related work; Section 5 presents our taxonomy development and validation; Section 6 describes our experimental methodology; Section 7 reports our empirical findings on the compositional gap and neural affinity ceiling effect; Section 8 discusses implications and limitations; and Section 9 concludes.

2. A Primer on Neural Affinity and the Transformer Architecture

This section synthesizes the architectural foundations that motivate our taxonomy. Comprehensive literature citations follow in Section 4.

2.1. The Transformer's Core Trade-Off: Global Reach vs. Local Blindness

The Transformer's central innovation, self-attention (Vaswani et al. 2017), provides a global receptive field from its first layer by establishing a fully connected graph where the path length between any two tokens is constant. This design grants the architecture an intrinsic affinity for tasks defined by long-range context. This capability manifests as specific, interpretable mechanisms, such as the global "association" attention pattern, where heads learn to link all occurrences of an identical token based on content, irrespective of position (Zhang et al. 2022).

This power, however, is predicated on a foundational trade-off: in exchange for powerful global aggregation, the architecture discards the strong local inductive biases inherent to models like CNNs. The resulting design tension is starkly evident in grid-based domains like ARC, where vanilla Vision Transformers (ViTs) fail catastrophically until explicit 2D-aware structural priors are re-introduced to ground the global attention mechanism (Li et al. 2024). Even on synthetic reasoning tasks, successful models must evolve distinct attention heads for global "association" versus local "manipulation," demonstrating that local processing is a learned compensation, not a native capability (Zhang et al. 2022). This highlights a fundamental dilemma for grid-based reasoning tasks like ARC: the Transformer's power in global aggregation is predicated on its architectural indifference to the local, structured feature synthesis required for spatial problem-solving. As further evidenced by analyses that interpret attention heads as implicit graph processors, structural priors can remediate this tension by providing relational scaffolds when local structure is essential (Sanford, Hsu, and Veličković 2024).

2.2. Algorithmic Deficiencies: The Mismatch with Symbolic & Sequential Reasoning

This architectural trade-off leads to fundamental algorithmic deficiencies when tasks require symbolic precision or sequential state management—both hallmarks of ARC. A Transformer with a fixed depth of `L` layers is limited to approximately `L` parallelizable computational steps, rendering it architecturally unsuited for algorithms requiring a dynamic or unbounded number of iterations (Chen and Zou 2024). This is not a theoretical curiosity but an empirical reality. Mechanistic analysis of Transformers trained on graph search tasks reveals they do not learn iterative algorithms like Depth-First Search. Instead, they converge on a brittle, fixed-depth parallel "path-merging" algorithm that fails to generalize as problem size and required search depth grow (Saparov et al. 2024). The model does not learn to iterate; it learns a shallow simulacrum of the algorithm that breaks on longer problem instances.

This inability to execute robust, multi-step procedures is compounded by architectural features that disrupt symbolic precision—especially for counting. In particular, `softmax` normalizes attention weights to sum to one, erasing absolute magnitude (e.g., two vs. five items), and `LayerNorm` rescales activations, further disturbing precise numerical values (Ouellette et al. 2023). See also (Behrens et al. 2025), which isolates how attention and feed-forward layers interplay to enable or hinder counting. The RASP-Generalization

TABLE 1. Theoretical neural affinity levels and associated task primitives

Affinity Level	**Associated Task Primitives**
High	Local color/pattern ops; global spatial transforms
Medium	Multi-step composition of known primitives
Low	Graph-like/topological reasoning (S3); iterative (A1)
Very Low	Combinatorial search (A2)

Conjecture offers a unifying framework for these failures, positing that Transformers are unlikely to learn inherently serial algorithms—from graph search to parity checking—that lack short, parallelizable programs in the Transformer's native computational model of selection and aggregation (Zhou et al. 2023). Relatedly, attempts to learn iterative cellular dynamics further highlight fixed-depth limitations (Burtsev 2024). This demonstrates that the deficiency is not one of learning capacity, but a fundamental mismatch between classical serial algorithms and the Transformer's native computational primitives.

2.3. Synthesizing the Transformer's Theoretical Neural Affinity for ARC

This body of evidence reveals a clear hierarchy of neural affinity for the computational primitives found in ARC, which we formalize into four levels:

In this hierarchy, **High Affinity** corresponds to tasks solvable via global pattern matching and information aggregation, leveraging the Transformer's native associative retrieval capabilities; **Medium Affinity** captures tasks that require multi-step composition of these primitives through the model's layered structure, introducing moderate difficulty; **Low Affinity** comprises tasks demanding precise local manipulation, graph-like reasoning, or symbolic precision—domains hindered by architectural components like `softmax` and `LayerNorm`; and **Very Low Affinity** denotes tasks requiring iterative algorithms or combinatorial search, which are fundamentally limited by the model's fixed-depth, non-recurrent architecture.

This theoretical hierarchy, summarized in the table below, motivates our 9-category taxonomy and the Neural Affinity Framework. It allows us to map specific ARC task categories to their expected difficulty for a standard Transformer, providing a predictive lens through which to analyze our empirical results.

With this theoretical framework for architectural affinity established, we now turn to the specific problem domain where we will test its predictive power: the Abstraction and Reasoning Corpus.

3. Background

3.1. The Abstraction and Reasoning Corpus (ARC): A Benchmark for Fluid Intelligence

The Abstraction and Reasoning Corpus (ARC) (Chollet 2019) was introduced not merely as another benchmark, but as a direct challenge to the prevailing methods of evaluating artificial intelligence. Its design is deeply rooted in a philosophical framework articulated in "On the Measure of Intelligence" (Chollet 2019), which posits that true intelligence cannot be measured by skill on known tasks, but must instead be gauged by the **efficiency of skill acquisition in the face of novelty**.

Chollet argues that much of the perceived progress in AI has resulted from "buying skill" with massive datasets and computational power, a process reliant on interpolation within a known data distribution rather than genuine, flexible reasoning. To counteract this, ARC was designed to be trivial for humans—who possess strong, generalizable priors—but exceptionally difficult for AI systems that lack these foundational abilities. The benchmark's core mechanics are engineered to enforce this distinction:

- **Few-shot learning.** Each task provides only a handful of demonstration pairs (typically 2–5). This data-sparse setting renders methods based on statistical pattern matching ineffective, forcing abstraction and rule induction.
- **Novelty.** The evaluation set contains tasks that are unique and unknown in advance, a critical control that measures a system's ability to handle "unknown unknowns" rather than performance on a familiar distribution (Chollet 2019, p. 45).
- **Core Knowledge priors.** Tasks are solvable using intuitive priors humans are believed to possess innately, such as *objectness* (parsing a scene into discrete objects), *goal-directedness*, *numbers and counting*, and *basic geometry and topology* (e.g., symmetry, connectivity) (Chollet 2019, p. 47–50). Failures on ARC often indicate missing fundamental world-models.
- **Algorithmic precision.** The pixel-perfect output requirement transforms the task from plausible generation into precise, algorithmic execution. A solver cannot merely produce an output that "looks right"; it must implement the inferred transformation exactly.

This design makes ARC a powerful instrument for measuring a system's **fluid intelligence**—its ability to reason and solve novel problems independent of acquired knowledge. The persistent gap between human performance and that of even the largest AI models highlights a fundamental architectural misalignment. Our work directly engages with this challenge by operationalizing Chollet's framework: we use `re-arc` to forge the necessary **priors**, we use our taxonomy to deconstruct the **generalization difficulty** of ARC tasks, and we use fine-tuning experiments to measure **skill-acquisition efficiency**. This methodology allows us to systematically diagnose the sources of this architectural misalignment.

3.2. The re-arc Dataset and Hodel's Calls to Action

The few-shot constraint of the original ARC-AGI-1 benchmark, while effective for testing human-like generalization, made large-scale controlled experiments infeasible. Hodel et al. (2024) addressed this limitation by introducing re-arc, a procedurally generated dataset containing unlimited examples for all 400 ARC training tasks. By reverse-engineering the transformation logic—what Hodel terms the "spirit" of each task—into a programmatic generator, re-arc transforms ARC from a static benchmark into a dynamic experimental testbed, enabling "fundamental scientific experiments" that were previously impossible.

Hodel's paper articulates a research program through four specific calls to action, each of which this work systematically addresses:

1. **Conduct within-task generalization experiments.** Hodel proposes "comparing various model architectures...in a setting of having a separate model trained for each task: For a given fixed number of examples, how well can the model generalize to an unseen set of examples of the same task?" [Hodel et al., 2024, Section 1]. We directly implement this experiment at scale: our task-specific fine-tuning protocol (Section 6.4) trains 186 individual models via LoRA on 400 examples each and evaluates on held-out examples, measuring within-task generalization across the full task distribution.

2. **Gather important insights into how progress could be made.** While acknowledging that within-task experiments might not yield "strong statements about suitability of approaches," Hodel argues they "may [be] a means to gather important insights" [Hodel et al., 2024, Section 1]. We embrace this diagnostic philosophy: rather than pursuing state-of-the-art scores, we identify the Compositional Gap as the primary failure mode and the Neural Affinity Ceiling Effect as its mechanism, providing actionable insights for architectural innovation (Sections 7-8).

3. **Enable non-uniform curricula construction.** Hodel explicitly calls for improving sample efficiency through "constructing a curriculum that is not fully random or uniform in terms of some metric" [Hodel et al., 2024, Section 1], even proposing preliminary metrics like RNG-Difficulty and PSO-Difficulty as "inspiration" [Hodel et al., 2024, Section 5]. Our 9-category taxonomy and Neural Affinity Framework provide a principled, empirically validated metric system for this purpose, with dual validation at both the code level (97.5% accuracy) and the visual level (36.25% overall 9-way accuracy; 95.24% on S3). Our curriculum analysis (Section 5.5) then reveals that 35.3% of re-arc falls into low-affinity categories, providing the empirical foundation for stratified curriculum design (Section 8.2).

4. **Define task relatedness for across-task generalization.** Hodel distinguishes within-task generalization from "the much more relevant notion of across-task generalization: Can a model also solve different (but related) tasks?" [Hodel et al., 2024, Section 5], implic-

itly calling for a formal definition of "relatedness." Our taxonomy is the first empirically validated answer to this call, defining relatedness through shared computational primitives. Following Hodel's principle of using verifiers to establish credibility [Hodel et al., 2024, Section 4], we apply dual validation—first at the code level, then at the visual level—demonstrating generalizability by successfully labeling the human-designed ARC-AGI-2 benchmark (Section 5.3).

Section 8 provides a detailed discussion of how each contribution fulfills these calls and advances Hodel's vision for systematic, scientific progress on ARC.

4. Related Work

4.1. Transformer Architectural Limitations

A model's performance ceiling on any given task is fundamentally constrained by its architectural suitability for the computational primitives required by that task. Our Neural Affinity Framework provides a lens to analyze this constraint. This section justifies our affinity ratings by synthesizing evidence from the literature to argue for three critical weaknesses in the standard Transformer architecture, which directly correspond to the low-affinity categories in our taxonomy: (1) a misalignment with structured spatial and graph-based reasoning (S3), (2) an inability to learn robust, iterative algorithms (A1/A2), and (3) architectural barriers to symbolic precision and counting (L1).

4.1.1. Weaknesses in Spatial and Graph-based Reasoning (Justifying S3's Low Affinity)

The Transformer's core design trades local inductive biases for a global receptive field, making it inherently ill-suited for tasks demanding precise local and topological reasoning. The architecture's central innovation, self-attention, grants it a global receptive field from its first layer by establishing a fully connected graph over its input tokens (Vaswani et al., 2017). However, this comes at a steep price: the forfeiture of the strong, built-in local processing capabilities of architectures like CNNs. As a deep mechanistic analysis by Raghu et al. (2021) reveals, Vision Transformers (ViTs) must learn local information processing from scratch, a capability that is hard-coded into CNNs and only emerges with massive-scale pre-training data. This makes local reasoning a data-hungry, non-native capability for the Transformer.

This architectural "blindness" to local structure leads to catastrophic empirical failures on abstract reasoning tasks like ARC. Li et al. (2025) provide direct evidence of this, showing that a vanilla ViT **"fails dramatically on most ARC tasks even when trained on one million examples per task,"** achieving a mean solve rate of only **17.68%**. They attribute this failure to an "inherent representational deficiency" that prevents the model from capturing the "spatial relationships between the objects" and handling "complex visual structures"—the very essence of topological reasoning. This confirms that the Transformer's global aggregation mechanism is misaligned with the demands of local, grid-based problem-solving, providing a firm justification for the Low Affinity rating of our S3 (Topological) category.

This failure at the pixel-feature level suggests a deeper representational challenge, which the success of hybrid neuro-symbolic systems confirms. For instance, the ARGA framework (Xu et al., 2022) succeeds by first converting ARC grids into an explicit "graph of objects with spatial or other relations" and then performing a combinatorial search. By reframing the problem in a more natural, symbolic space, ARGA solves complex object-centric tasks with **300 times fewer search nodes** than a leading Kaggle solution, suggesting that a graph-based abstraction is a more effective primitive for this reasoning domain than the Transformer's fully-connected attention graph over pixels.

This general difficulty with graph-like structures, however, is not uniform. As we will show in Section 7.4, it is most pronounced in tasks requiring true graph traversal and relational reasoning (our S3-B subcategory), while tasks with simpler, pattern-based topology (S3-A) prove more tractable for the Transformer architecture.

4.1.2. Inability to Learn Iterative and Search-based Algorithms (Justifying A1/A2's Very Low Affinity)

The Transformer's fixed-depth, non-recurrent architecture imposes a hard ceiling on its ability to perform tasks requiring a dynamic or unbounded number of sequential steps, such as iterative refinement or combinatorial search. Theoretically, a Transformer with L layers is limited to performing approximately L parallelizable computational steps in a single forward pass (Chen & Zou, 2024). This makes it architecturally unsuited for algorithms that are inherently serial or require an unknown number of iterations to converge on a solution, as is common in our A1 (Iterative Dynamics) and A2 (Spatial Packing) categories, justifying their "Very Low" affinity rating.

This theoretical limit is confirmed with stark empirical and mechanistic evidence. When trained on a canonical graph search task, the Transformer does not learn an iterative algorithm like Depth-First Search. Instead, Saparov et al. (2025) find that it converges on a brittle, fixed-depth parallel "path-merging" algorithm that fails to generalize as the required search depth increases. Crucially, they found that increasing model parameters from **0.9M to 60.4M did not resolve this scaling failure**, proving it is an architectural bottleneck, not a matter of capacity. This provides a direct mechanistic explanation for the hard performance ceilings observed in tasks requiring combinatorial search: the model fundamentally fails to learn an iterative process.

This fundamental limitation is so widely recognized that successful applications of Transformers to iterative problems consistently rely on external scaffolding to impose a sequential loop. To solve Constraint Satisfaction Problems (CSPs) like Sudoku, which typically require **20 to 60 steps of reasoning** (Yang et al., 2023), the standard Transformer is insufficient. Success is only achieved by either making the architecture explicitly *recurrent* (Yang et al., 2023) or by deploying a single-step "solution refiner" *iteratively* at test time (Xu et al., 2025). This workaround pattern is definitive proof that the base architecture lacks the necessary iterative capability. The state-of-the-art on ARC itself provides the ultimate validation for this architectural deficit: the 2024 ARC Prize report reveals that all top-scoring solutions were compelled to abandon the single-pass, static inference paradigm, instead relying on methods that externalize iterative reasoning. Discrete program search performs explicit algorithmic steps, while Test-Time Training iterates via gradient-based refinement, proving that success on ARC requires a mechanism for iteration that the base Transformer architecture simply does not possess (Chollet et al., 2024).

4.1.3. Architectural Barriers to Symbolic Precision and Counting (Justifying L1's Low Affinity)

Finally, specific components of the Transformer architecture are fundamentally at odds with preserving and manipulating precise quantitative information, as required for symbolic tasks like counting. These are not emergent weaknesses but direct consequences of core design choices. Ouellette et al. (2023) provide a definitive mechanistic analysis, identifying `softmax` and `LayerNorm` as the primary culprits. The `softmax` function, applied to attention scores, normalizes weights to sum to one, thereby erasing absolute quantitative information and making it impossible to distinguish attending to two items versus five. `LayerNorm` further **"disturbs the propagation of precise numerical values by rescaling activations"** based on the statistical properties of each activation vector, again destroying absolute magnitude.

This architectural incompatibility means that counting is not a native capability. When Transformers do learn to perform a counting-like task, they must resort to complex, non-obvious workarounds. Behrens et al. (2025) show that even a simple histogram task requires a **"delicate interplay between attention and feed-forward layers,"** where the model learns one of two specific strategies: a "relation-based" approach using attention for pairwise comparisons or a more parameter-heavy "inventory-based" approach using the FFN as a key-value memory. The fact that a seemingly simple task requires such complex, learned machinery—which is highly sensitive to architectural hyperparameters—reinforces the conclusion that counting has a low native affinity for the Transformer architecture. This fundamental barrier to precise cardinality representation naturally extends to set-theoretic operations like intersection, union, and difference, which rely on the accurate accounting of elements, thus justifying the "Low" affinity rating for our L1 (Logic/Set) category.

TABLE 2. Neural Affinity Ratings and Their Architectural Foundations

Category	Affinity	Architectural Limitation	Mechanism & Key Failure Mode	Primary Citations
A1, A2 (Iterative & Search)	**Very Low**	**Fixed-Depth, Non-Recurrent Architecture:** The model's depth imposes a hard limit on the number of sequential operations it can perform in a single forward pass.	**Inability to Learn Iterative Algorithms:** The model is bounded by its depth (*L* layers ≈ *L* parallel steps). Instead of a true, generalizable iterative process like DFS, it learns a brittle, fixed-depth "path-merging" algorithm—a shallow simulacrum of search that fails to scale with problem size.	Saparov et al. (2025), Chen & Zou (2024), Yang et al. (2023), Xu et al. (2025)
S3 (Topological & Graph)	**Low**	**Absence of Local Inductive Biases:** The architecture's global self-attention mechanism discards the strong, built-in local processing capabilities of models like CNNs.	**"Local Blindness" and Representational Deficiency:** The model struggles to model precise "spatial relationships between the objects." It must learn local processing from scratch, a data-hungry process that often fails on "complex visual structures" where an explicit "graph of objects" representation is more effective.	Li et al. (2024)†, Raghu et al. (2021), Xu et al. (2022)
L1 (Counting & Logic)	**Low**	**Destructive Normalization Components:** Core architectural layers are fundamentally at odds with preserving absolute quantitative information.	**Erasure of Absolute Magnitude:** `softmax` converts counts into a probability distribution, erasing absolute quantitative information. `LayerNorm` further "disturbs the propagation of precise numerical values by rescaling activations." This forces the model into brittle, learned workarounds instead of performing robust counting.	Ouellette et al. (2023), Behrens et al. (2025)

4.2. The State of the Art on ARC: The Scale & Scaffolding Paradigm

The current state-of-the-art on the Abstraction and Reasoning Corpus (ARC) is defined not by a single, monolithic model, but by a paradigm of **"scale and scaffolding."** This approach leverages large, pre-trained language models (8B+ parameters) but implicitly acknowledges their inherent architectural limitations for abstract reasoning. Success is achieved by augmenting these models with external, test-time processes that guide, constrain, or even replace the model's own reasoning process. The **ARC Prize 2024 Technical Report (Chollet et al., 2025)** provides the definitive validation for this paradigm, identifying these scaffolds—primarily Test-Time Training and program synthesis—as the key drivers behind the recent leap in SOTA performance. The report's stark finding that "there does not exist any static inference-style transduction solution that scores above 11%" proves that monolithic, single-pass Transformer models fail and that external reasoning support is, for now, a prerequisite for success.

4.2.1. Test-Time Training (TTT) as a Dynamic Scaffold

A primary strategy for overcoming the static limitations of pre-trained models is Test-Time Training (TTT). This technique functions as a dynamic scaffold by temporarily fine-tuning the model on a specific task's demonstration examples at the moment of inference. **Akyürek et al. (2024)** provide the seminal work in this area, demonstrating that applying TTT to an 8B-parameter model can yield up to a **6-fold increase in accuracy** on ARC compared to an identical fine-tuned baseline without it. This highlights a crucial insight: the base model possesses the necessary latent knowledge, but it requires an explicit, task-specific optimization process at test time to reconfigure that knowledge into a correct solution. The winners of the ARC Prize 2024, **Franzen et al. (2024)**, further validated this approach, making heavy use of TTT in their winning solution. TTT is therefore a form of *just-in-time specialization*—a powerful scaffold that aggressively adapts a generalist model to overcome its low native affinity for a specific, novel task.

4.2.2. Program Synthesis and Search as an External Reasoning Engine

An alternative and often complementary scaffolding approach offloads the reasoning process entirely from the neural network's implicit weights to an explicit, verifiable program. In this paradigm, the LLM acts as a high-level hypothesis generator, producing code that, when executed by an external interpreter, produces the final solution.

Wang et al. (2024) demonstrate the power of this externalization with their "Hypothesis Search" method. They use GPT-4 to first generate abstract hypotheses in natural language and then translate those hypotheses into verifiable Python programs. This multi-step process of explicit, structured reasoning significantly outperforms directly prompting the model for a solution (30% vs. 17%), proving that decomposing and externalizing the reasoning task is critical for success.

The state-of-the-art system from **Li et al. (2024)** formalizes this divide by combining an "inductive" program synthesis arm with a "transductive" direct-prediction arm (which itself relies heavily on TTT). Their core finding is that these two approaches are **"strongly complementary,"** with "inductive program synthesis excelling at precise computations... while transduction succeeds on fuzzier perceptual concepts." This observed complementarity is a central mystery in modern ARC research and creates the very "diagnostic gap" our work aims to fill by providing a systematic explanation for which tasks align with which paradigm.

4.3. Architectural Innovation as an Alternative Path

In contrast to the dominant paradigm of scaffolding massive, general-purpose models, an alternative research direction pursues **architectural innovation** as the primary path to progress. This approach posits that superior reasoning can be achieved not through external processes, but through more principled, often smaller, model designs trained from scratch. This methodology directly validates our own choice to focus on a small-scale, 1.7M parameter model to isolate and diagnose architectural properties, seeking to solve reasoning deficits through *internalization* (better architectures) rather than *externalization* (scaffolds).

4.3.1. Small, Specialized Transformers Trained from Scratch

The "Mini-ARC" project by **Fletcher-Hill (2024)** serves as a powerful existence proof for this architecture-first philosophy. Using a custom **67M parameter Transformer** trained exclusively on ARC data, he achieves a remarkable 41.2% accuracy on a constrained subset of the benchmark. Crucially, this result is achieved "with a very small model and without the use of search, language models, or program synthesis." Fletcher-Hill's work demonstrates that massive scale and extensive pre-training are not prerequisites for success on ARC. Furthermore, his findings provide strong external validation for our "Compositional Gap" thesis; he reports **95.3% cell-level accuracy but only a 41.2% grid-level solve rate**, a clear signal of local pattern mastery without successful global synthesis.

4.3.2. Novel Architectures with Built-in Adaptation

Pushing architectural innovation further, **Bonnet & Macfarlane (2024)** introduce the Latent Program Network (LPN), a novel 39M parameter architecture trained from scratch on ARC tasks. Instead of applying adaptation externally via TTT, the LPN builds the search mechanism *directly into the architecture*. It learns a continuous latent space of "programs" and performs a gradient-based search within this representational space at test time. This represents a fundamental shift from adapting a model's *parameters* (as in TTT) to searching over its *representations*. Their work not only validates the small-model approach but also demonstrates that core reasoning processes like search can be internalized through novel architectural design. Their success challenges the notion that vision transformer architectures are fundamentally limited on ARC, suggesting instead that reported bottlenecks may reflect paradigm constraints (one-shot generalist approaches) rather than strict architectural limitations when specialist training and built-in search mechanisms are employed.

4.4. Our Contribution in Context: The Missing Diagnostic Framework

The current ARC literature presents a fractured landscape. The state-of-the-art (Section 4.2) is dominated by complex scaffolds applied to massive models, while a promising alternative path (Section 4.3) explores smaller, bespoke architectures. This leaves a critical **diagnostic gap**: the community lacks a systematic methodology to explain *why* scaffolding is so effective, *why* different methods are complementary, and *where* targeted architectural innovation is most urgently needed. Our work fills this gap by providing the first comprehensive diagnostic toolkit for ARC, transforming ad-hoc engineering into a principled science.

First, our empirical discovery of the **"Compositional Gap"** provides the mechanistic explanation for *why scaffolding works*. The stark dissociation between a model's ability to learn local patterns (high cell-level accuracy) and its inability to synthesize a globally correct solution (low grid-level accuracy) is the primary failure mode of standard Transformers. Scaffolding methods are effective precisely because they bridge this gap: program synthesis (**Li et al., 2024; Wang et al., 2024**) externalizes the difficult compositional step into a verifiable, executable program, while TTT (**Akyürek et al., 2024**) aggressively specializes the model at test time, temporarily forcing it to overcome its compositional limitations for a single, low-affinity task.

Second, our **Neural Affinity Framework** and **9-category taxonomy** provide the formal language to explain the striking *complementarity* observed by **Li et al. (2024)** and **Bober-Irizar & Banerjee (2024)**. The low overlap between program synthesis and direct neural methods—as low as 37% in one study—is not a random artifact; it is a predictable outcome of their differing architectural affinities. Our framework allows us to map their qualitative descriptions to concrete task categories: program synthesis excels on tasks with low neural affinity for Transformers, such as those requiring precise iteration or logic (our `A1`, `A2`, and `L1` categories), while neural methods succeed on tasks with high affinity, such as those rooted in perception and pattern transformation (our `C1` and `S3-A` categories).

Finally, our framework provides a *generative roadmap* for architectural innovation. While diagnostic work like **Mitchell et al. (2023)** confirms that even SOTA models like GPT-4 lack "robust abstraction abilities at humanlike levels," our framework specifies *where* these failures are most acute (e.g., in low-affinity categories like `A2: Spatial Packing` and `S3-B: Graph-like Reasoning`). This allows researchers to move from the general goal of "better reasoning" to the specific, targeted goal of designing affinity-aligned modules for known architectural bottlenecks, guiding the work of innovators like **Fletcher-Hill (2024)** and **Bonnet & Macfarlane (2024)** toward the problems that need them most. In essence, our work provides the map and compass that the ARC community needs to navigate from scaling and scaffolding toward truly intelligent, compositional architectures.

5. A Systematic Taxonomy of the ARC Problem Space

To systematically analyze the architectural limitations of Transformers on abstract reasoning, we first require a formal language to describe the problem space itself. Prior work on the Abstraction and Reasoning Corpus (ARC) has lacked a comprehensive, empirically validated taxonomy, making it difficult to move beyond aggregate performance metrics to a principled diagnosis of model failures. In response, this section introduces a 9-category taxonomy of the `re-arc` problem space. We detail a rigorous dual-validation methodology, first at the code level and then at the visual level, to establish its scientific validity. We then use this framework to reveal a significant curriculum bias in `re-arc` and formalize the Neural Affinity Framework that serves as the theoretical lens for the remainder of this paper.

5.1. A 9-Category Framework for Abstract Reasoning

Our taxonomy is built on a two-dimensional theoretical basis, classifying tasks by their **Primary Transformation Type** (what property of the grid changes) and their **Reasoning Mode** (how the change is computed). This structure yields nine distinct categories that group tasks requiring similar computational primitives.

TABLE 3. The 9-Category ARC Taxonomy and `re-arc` Curriculum Distribution

Category	Name	Description	Count	% of Total
S3	Spatial Topology	Reasoning about connectivity, paths, neighbors, and graph-like structures.	108	27.0%
C1	Color Transform	Recoloring objects or regions without changing their spatial structure.	99	24.8%
S1	Spatial Local	Single-step spatial rearrangements like mirroring, rotation, and direct shifting.	52	13.0%
S2	Spatial Global	Multi-step spatial compositions, such as tiling, concatenation, or replication.	38	9.5%
C2	Color Pattern	Applying colors based on a template or complex pattern-matching logic.	28	7.0%
A2	Packing	Placing objects onto a grid under spatial or geometric constraints (e.g., no overlaps).	28	7.0%
L1	Logic Set	Applying set-theoretic operations like intersection, union, and difference.	21	5.2%
K1	Scaling Operations	Changing grid or object size via `upscale`, `downscale`, or `crop` primitives.	7	1.8%
A1	Iterative	Repeatedly modifying a grid's state until a convergence condition is met.	5	1.2%
Ambiguous	—	Temporarily ambiguous to the rule-based classifier due to unhandled DSL primitives or mixed cues; manually resolvable into the 9 categories (see §5.4).	14	3.5%
Total			**400**	**100.0%**

5.1.1. The Two Dimensions

Primary Transformation Type

What property of the grid changes.

- **Spatial (S1, S2):** Grid-level geometric rearrangements.
- **Topological (S3):** Connectivity, paths, graph-like operations.
- **Chromatic (C1, C2):** Color-based transformations.
- **Cardinality/Scaling (K1):** Size and resolution changes.
- **Logical/Set (L1):** Set-theoretic operations.
- **Algorithmic (A1, A2):** Iterative or search-based procedures.

Reasoning Mode

How the change is computed.

- **Direct:** Single-step application (e.g., S1, C1, K1).
- **Compositional:** Multi-step combinations (e.g., S2).
- **Iterative:** Loop-based refinement (A1).
- **Search-based:** Constraint satisfaction (A2).
- **Pattern-based:** Template matching and application (C2).

The complete taxonomy and the distribution of the 400 `re-arc` tasks are presented in Table 3.

Note: The canonical definitions for each category and the full theoretical basis are detailed in the taxonomy specification (see Appendix B).

5.2. Methodology: A Dual Validation and Generalization Framework

To ensure our taxonomy is both scientifically robust and relevant to the visual nature of ARC, we developed a two-stage validation framework. First, we validated the taxonomy at the code level using a rule-based classifier on the `re-arc` generator code. Second, we validated its visual coherence by training a CNN to recognize these categories from raw grid pixels.

5.2.1. Stage 1: Code-Based Validation with a Rule-Based Classifier

We developed a rule-based classifier through a rigorous **21-iteration process**, analyzing the `re-arc` generator code to identify patterns of DSL primitive usage. This iterative refinement involved not only adding rules but also discovering and correcting 4 ground truth errors in our initial manual labels, a testament to the process's rigor. A key breakthrough was the implementation of an "Execution Order" heuristic, which distinguishes between primitives used for primary transformations versus those used for setup or packaging by analyzing *when* they occur in the code.

The validation of this classifier was twofold: 1. On our 40-task validation set, the final classifier correctly categorized 39 of the 40 tasks, for an accuracy of **97.5%**. When applied to the full 400-task dataset, the classifier successfully categorizes **386 tasks**, with 10 misclassifications among the 396 classifiable tasks (yielding **97.5%** accuracy on classifiable tasks) and 14 tasks (3.5%) as ambiguous due to unhandled edge-case primitives. The 40-task validation set was pre-defined prior to the final rule iterations and includes representation from all major categories. 2. To verify unbiased performance and ensure our validation set wasn't accidentally biased toward easy cases, we tested the final classifier on a held-out, stratified random sample of 8 tasks (one per major category) it had never seen during development. On this "final exam," the classifier achieved **100% accuracy (8/8)**, confirming that the taxonomy generalizes reliably beyond the development set.

This dual validation provides high confidence that our taxonomy is a reliable and generalizable representation of the `re-arc` problem space at the code level.

Interpretive Note: As the reviewer Hodel would rightly point out, our rule-based classifier achieves 97.5% accuracy in capturing *his* implementation patterns. This validates the taxonomy's self-consistency within the `re-arc` codebase but does not, by itself, prove these categories represent universal primitives independent of his coding choices. The visual validation and ARC-AGI-2 transfer experiments (Section 5.3) provide the critical evidence that these implementation-level categories correspond to visually coherent and generalizable reasoning patterns.

Classification Methodology: Our classifier analyzes generator code by (1) extracting the final transformation window (last 15 lines before `return`), (2) identifying DSL primitives (functions like `box()`, `fill()`, `crop()`), and (3) applying priority-ordered classification rules. The key design principle is **priority ordering**: topological operations (S3) are detected before iterative patterns (A1/A2) to prevent misclassification when both patterns appear in the same generator. For example, a task with both `box()` (topological) and `while` (iterative) is classified as S3 because the topological primitive represents the primary transformation. Complete decision rules, worked examples, and visual signatures are provided in **Appendix A: Practical Classification Guide**, enabling other researchers to classify new tasks without running our classifier code.

5.2.2. Stage 2: Visual Validation with a CNN Classifier

To confirm that our code-based categories correspond to learnable visual patterns—a critical step to bridge the gap to the purely visual ARC-AGI-2 benchmark—we trained a convolutional neural network (TaskEncoderCNN) to classify tasks from raw grid demonstrations. For each task, the model processes 3 demonstration examples. The architecture first embeds discrete grid values (0-10) into continuous vectors using a learned embedding layer, concatenates input and output embeddings for each demonstration, then processes the concatenated representation through convolutional layers (depth=6, embed_dim=512) followed by global average pooling. Features from all demonstrations are aggregated and passed through an MLP to produce the final task embedding, which is compared against category centroids derived from the champion model's task-specific LoRA adapters (averaged within each category). The architecture and hyperparameters (learning rate, weight decay, label smoothing) were determined through a systematic 150-trial HPO sweep using Optuna. Training used cross-entropy loss with an 80/20 split of the 400 `re-arc` tasks and employed cosine annealing for learning rate scheduling. Full implementation details are provided in the reproduction package (see Appendix B).

In the full 9-way classification task, the CNN reached an overall validation accuracy of **36.25%** (random baseline **11.1%**), representing a **3.3**× improvement over chance. This exceeds the typical benchmark for visual category learning on small datasets (2-3× over chance), providing strong evidence that the model learns meaningful category structure despite the difficulty of distinguishing all 9 categories. Notably, the model's per-category accuracy on **S3 (Spatial Topology)** was **95.24%**, indicating this category exhibits a particularly consistent and learnable visual signature that the CNN reliably recognizes within the 9-way task. This strong performance on S3 (the curriculum's largest category at 27%) provides empirical validation that our code-based taxonomy successfully captures visually coherent task groups. Given the small sample sizes for some validations (e.g., 40-task and 8-task sets), we report point estimates and acknowledge the resulting uncertainty; Section 7 provides larger-sample analyses that corroborate these findings.

5.3. Generalizing the Taxonomy to ARC-AGI-2

The validated CNN classifier was then employed as a diagnostic tool to apply our taxonomy labels to the human-designed tasks in the ARC-AGI-2 benchmark. For each of the 1120 ARC-AGI-2 tasks, its demonstration grid pairs are fed into the trained CNN, which outputs a predicted category label. This process enables the first systematic, category-based analysis of the generalization gap between `re-arc` and ARC-AGI-2, the results of which are presented in Section 7.3. The inference script is included in our reproduction package (see Appendix B) to ensure this step is fully reproducible.

Methodological Caveat on Label Reliability: These category assignments for ARC-AGI-2 tasks are derived from our visual classifier, which achieves 36.25% overall 9-way accuracy (despite 95.24% on S3). For non-S3 categories, label uncertainty is substantial. We therefore focus our strongest claims on high-confidence S3 predictions and treat other category assignments as provisional diagnostic tools rather than ground truth labels. Future work should validate these labels via human expert annotation or alternative classification methods.

5.4. Characterizing the Limits: Analysis of Ambiguous Tasks

Our rule-based classifier designated 14 of the 400 `re-arc` tasks (3.5%) as "ambiguous." A manual analysis, summarized in Table 4, revealed that these tasks are not fundamentally uncategorizable. Instead, they represent systematic gaps in the classifier's rule set, primarily due to the use of edge-case DSL primitives (e.g., `subgrid()`, `crop()`, `switch()`) for which specific rules were not implemented. For example, three tasks that perform cropping operations were flagged as ambiguous because the classifier's K1 rules only check for `upscale()` and `downscale()`.

This analysis serves to define the "semantic ceiling" of our rule-based approach. The fact that all 14 ambiguous tasks can be manually assigned to one of the existing 9 categories reinforces the completeness of the taxonomy itself.

5.5. Finding 1: The `re-arc` Curriculum is Biased Toward Low-Affinity Tasks

Our analysis of the full 400-task `re-arc` curriculum reveals a significant distributional bias. As shown in Figure 1, the two largest categories are **S3 (Spatial Topology)** at 27.0% and **C1 (Color Transform)** at 24.8%. Most critically, we find that **35.3%** of the curriculum is composed of tasks with low or very low neural affinity for standard Transformers.

This figure is calculated from the final classification data (see Appendix B) by summing the counts for categories defined as Low (S3, A1) and Very Low (A2) affinity in the Neural

TABLE 4. Analysis of 14 Ambiguous Tasks

Task ID	Classifier Output	Correct Category	Reason for Classifier Failure
`0b148d64`	ambiguous	**K1**	Unhandled `subgrid()` operation used for cropping.
`1c786137`	ambiguous	**K1**	Unhandled `crop()` operation.
`d10ecb37`	ambiguous	**K1**	Unhandled `crop()` operation.
`b94a9452`	ambiguous	C2	Unhandled `switch()` operation for color swapping.
`c3e719e8`	ambiguous	S3	Complex position-based tiling pattern.
`a1570a43`	ambiguous	S2	Unhandled spatial displacement reversal.
`25d8a9c8`	ambiguous	L1	Unhandled row-wise conditional logic.
`5582e5ca`	ambiguous	A2	Missed `canvas(attribute)` pattern for attribute extraction.
`995c5fa3`	ambiguous	A2	Complex shape-to-color mapping pattern.
`cdecee7f`	ambiguous	A2	Complex attribute extraction and grid arrangement.
`d4469b4b`	ambiguous	A2	Complex background-color-to-pattern mapping.
`d631b094`	ambiguous	A2	Output dimension based on input cell count.
`6d0160f0`	ambiguous	A1	Pattern selection based on a marker attribute.
`cce03e0d`	ambiguous	S3	Complex pattern replication based on color location.

Source: See Appendix B for detailed analysis. Note the correction of cropping tasks to K1.

Affinity Framework detailed in Section 5.6:

$$\frac{(\text{S3} = 108) + (\text{A1} = 5) + (\text{A2} = 28)}{400} = \frac{141}{400} = 35.3\%$$

This distributional skew towards architecturally challenging tasks is a central finding that motivates our investigation into performance ceilings.

Data Availability: The complete task-to-category mapping is provided in our reproduction package (see Appendix B), enabling reproducible category-specific analyses throughout this work.

5.6. The Neural Affinity Framework

To connect our empirical findings to architectural theory, we formalize the **Neural Affinity Framework**. This framework maps each of the 9 taxonomy categories to one of four affinity levels, representing the inherent architectural suitability of a standard Transformer for that category's computational primitives.

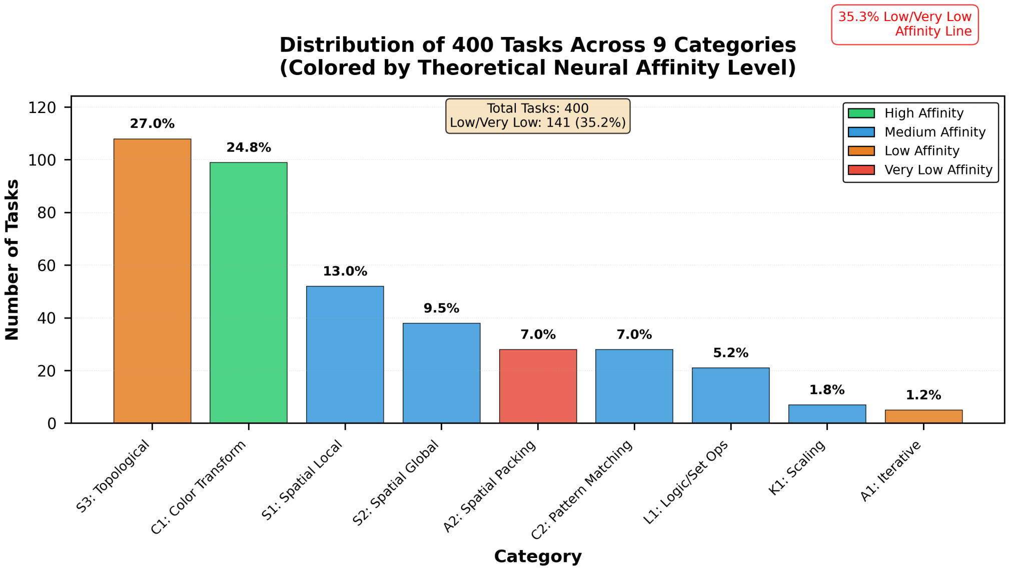

FIGURE 1. Bar chart showing the distribution of 400 `re-arc` tasks across 9 taxonomy categories, color-coded by Neural Affinity level (High: green, Medium: blue, Low: orange, Very Low: red). The horizontal dashed line at 35.3% marks the total proportion of Low and Very Low affinity tasks in the curriculum, demonstrating significant bias toward architecturally challenging primitives. S3 (Topological) is the dominant category at 27.0%, followed by C1 (Color Transform) at 24.8%. Categories are sorted by count (descending), with affinity levels indicated by color to reveal the curriculum's architectural challenge profile. See Appendix B for generation details.

5.6.1. Theoretical vs. Empirical Affinity

It is critical to distinguish between the two ways "affinity" is used in this paper, as they measure fundamentally different properties:

	Theoretical Affinity *(Table 5; curriculum-bias calculation)*	**Empirical Affinity** *(champion baseline analysis, Section 7)*
What it measures	Architectural suitability for a category's computational primitive (e.g., "Can attention mechanisms handle graph reasoning?")	Observed local pattern understanding, quantified by cell-level accuracy
Granularity	Category-level (9 categories)	Task-level (individual tasks)
Source	Theoretical framework (Section 2) and literature justification (Section 4)	Champion model's `base_cell_accuracy` performance
Thresholds	—	High (> 85%), Medium (70–85%), Low (< 70%)
Purpose	Predicts which task types should be challenging based on Transformer limitations	Measures whether the model learned to recognize local patterns within each cell

Critical distinction. Theoretical affinity predicts architectural difficulty based on computational primitives, while empirical affinity measures observed local pattern learning. These can diverge: a category may be theoretically challenging yet show high cell-level accuracy if the model learns to recognize local patterns without mastering global composition. Section 7 analyzes this divergence systematically across all categories, revealing that the primary failure mode is compositional, not perceptual.

5.6.2. The Four-Level Affinity Hierarchy

The mapping is justified by a powerful narrative loop: the theoretical principles of Transformer limitations (from Sections 2 and 4) predict which task types should be difficult, and the empirical curriculum distribution (Section 5.5) confirms that these very task types are overrepresented.

For example, **A1 (Iterative)** and **A2 (Packing)** are rated "Very Low" affinity because the standard Transformer's fixed-depth, non-recurrent architecture is fundamentally unsuited for algorithms requiring loops, stateful updates, or combinatorial search, as detailed in Section 2.2. The interpretive payoff of this framework is twofold: it provides a principled explanation for the curriculum bias discovered in Section 5.5, and it allows us to formulate the specific, testable predictions about architectural performance ceilings that we explore in Sections 7 and 8.

TABLE 5. The Neural Affinity Framework: Mapping Task Categories to Architectural Suitability

Affinity Level	**Code**	**Category Name**	**Justification**	**% of `re-arc` (N=400)**
High	C1	Color Transform	Color transformations are native operations for attention mechanisms.	24.8%
Medium	S1	Spatial (Local)	Local spatial operations benefit from position-aware embeddings.	13.0%
Medium	S2	Spatial (Global)	Multi-step composition (concat/tiling) is learnable via layers.	9.5%
Medium	C2	Color (Pattern)	Cross-attention between regions for template matching.	7.0%
Medium	K1	Scaling	Local aggregation for upscaling/downscaling operations.	1.8%
Medium	L1	Logic (Set Ops)	Pairwise element comparison for set operations.	5.3%
Low	S3	Spatial (Topology)	Graph-like reasoning (connect/extend) challenges fixed attention.	27.0%
Low	A1	Algorithmic (Iterative)	Iterative refinement requires recurrence, hard for feedforward.	1.3%
Very Low	A2	Algorithmic (Packing)	Combinatorial search fundamentally limited by fixed depth.	7.0%

Note: Percentages include all 400 tasks for consistency with the curriculum bias analysis (Section 5.5). The 14 ambiguous tasks (3.5%) are excluded from this framework, so percentages sum to 96.5% by design. Rounding accounts for small discrepancies. Justification for each mapping is grounded in the architectural analysis of Sections 2 and 4.

6. Experimental Methodology

This section describes our experimental approach for validating the ARC Taxonomy framework. All experiments employ a pre-trained "Champion" model that operationalizes Chollet's framework for measuring intelligence (Chollet 2019). We evaluate this model through three complementary experiments: (1) architectural ablation to validate design choices, (2) task-specific fine-tuning to measure skill-acquisition efficiency, and (3) out-of-distribution generalization to ARC-AGI-2 to validate the taxonomy's predictive power.

Critical Methodological Note: The initial champion checkpoint (see Appendix B) was trained using our internal `jarc_reactor` codebase, while all experiments reported here were conducted using a clean-room CS336-based reimplementation. This two-phase approach arose from discovering critical flaws in the original codebase (Context Starvation, Identity-Copy Bias) that necessitated a principled rebuild. Full justification and implementation details are provided in **Appendix C**.

6.1. The Champion Model: A Pre-trained Generalist

6.1.1. Provenance and Architecture

Our Champion model was selected through systematic hyperparameter optimization (HPO) on ARC-AGI-2 and subsequently pre-trained on synthetic `re-arc` tasks. This two-stage process operationalizes Chollet's framework: HPO creates an optimized architecture, then pre-training forges synthetic Core Knowledge priors (objectness, topology, geometry) before measuring skill-acquisition efficiency.

The Champion architecture is an encoder-decoder transformer comprising approximately 1.7M parameters. The design incorporates four key components: an encoder–decoder structure for sequence-to-sequence transformation, Grid2D positional encoding to capture spatial relationships, permutation-invariant color embeddings to handle arbitrary color mappings, and a context bridge mechanism to facilitate information flow between input examples and target predictions. All hyperparameters, including the learning rate (`lr=0.00185`) and maximum grid size (`max_grid_size=30`), were derived from Trial 69 of our systematic hyperparameter optimization sweep.

Implementation: Built on CS336 principles [Stanford CS336, 2024] with test-driven development (TDD) and explicit data contracts. The architecture is a systematic construction from a trusted academic baseline, not a monolithic black box. All code is open-source in the reproduction package.

6.1.2. Pre-training Protocol

Dataset: `distributional_alignment` (400 `re-arc` tasks), split 308 training / 92 validation (see Appendix B for details).

Training proceeded in two phases with different sample densities. The bootstrap phase utilized 15 samples per task, with the epoch 3 checkpoint achieving 2.34% grid accuracy; this checkpoint serves as our baseline for cross-benchmark comparisons due to its low-sample regime that matches realistic few-shot scenarios. The main analysis phase employed 150 samples per task, reaching peak grid accuracy of 3.25% at epoch 23 and peak cell accuracy of 81.80% at epoch 36.

Key Finding: Training dynamics revealed dissociation between grid and cell accuracy (Figure 2). Grid accuracy peaks early while cell accuracy continues improving, demonstrating that the model learns local patterns but struggles with global composition—the signature of the Compositional Gap analyzed in Section 7.

6.2. Architectural Ablation: Validating Design Choices

To validate our architectural choices, we conducted an ablation study using independent component testing. We evaluated five parameter-matched variants (approximately 1.7M

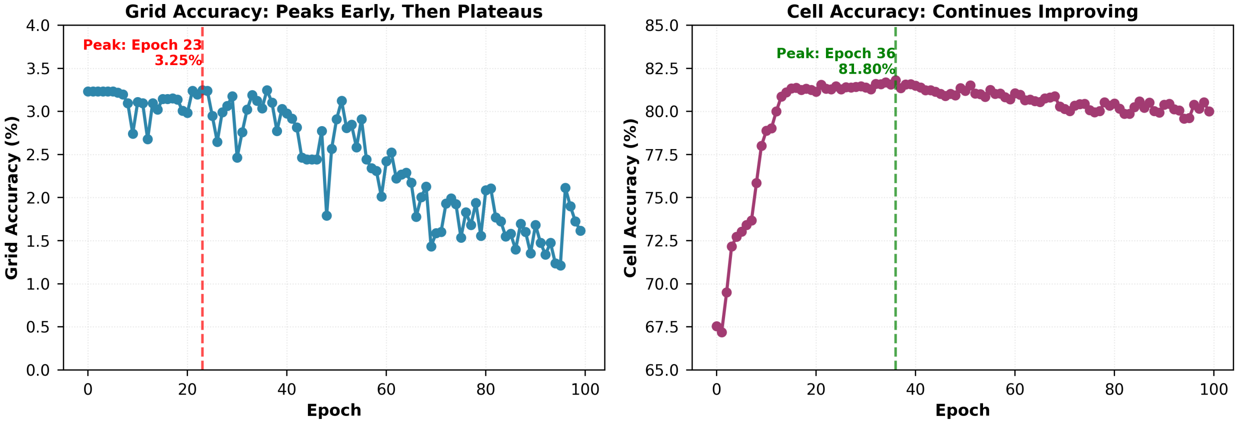

FIGURE 2. Grid-Cell Dissociation During Training. *Left panel:* Grid accuracy peaks at epoch 23 (3.25%) before plateauing. *Right panel:* Cell accuracy continues improving to epoch 36 (81.80%). The 13-epoch gap reveals the model learns local patterns but struggles with global composition.

± 2% parameters each): Exp-1 employed a pure decoder-only autoregressive baseline; Exp0 implemented a standard encoder-decoder architecture to serve as our baseline for comparisons; Exp1 augmented the baseline with Grid2D positional encoding only; Exp2 added permutation-invariant embeddings only; and Exp4 represents our full Champion architecture incorporating all components.

While ablation design limitations prevent strong claims about individual component contributions (see Appendix C for full analysis), the development process yielded two critical reproducibility findings. First, the encoder–decoder architecture provides superior training stability compared to decoder-only variants. Specifically, Exp-1 exhibited pathological training dynamics, peaking at epoch 1 before degrading, while all encoder-decoder variants maintained stable training curves through 100 epochs. Second, model performance demonstrates high sensitivity to the `max_grid_size` hyperparameter: an inadvertent mismatch (35 versus 30) caused training instability and 31% performance degradation. All experiments reported in this work employ the corrected value of `max_grid_size=30`. These findings both validate our encoder-decoder architectural choice and establish critical configuration requirements for reproducibility.

6.3. Experiment 1: Task-Specific Fine-Tuning via LoRA

To measure skill-acquisition efficiency and isolate architectural ceilings, we fine-tuned the Champion model on individual `re-arc` tasks. To make this large-scale experiment computationally feasible, we employed **Low-Rank Adaptation (LoRA)** [Hu et al., 2021], a parameter-efficient fine-tuning (PEFT) technique. Instead of retraining the full 1.7M parameters of the Champion model for each task, LoRA freezes the base model and injects

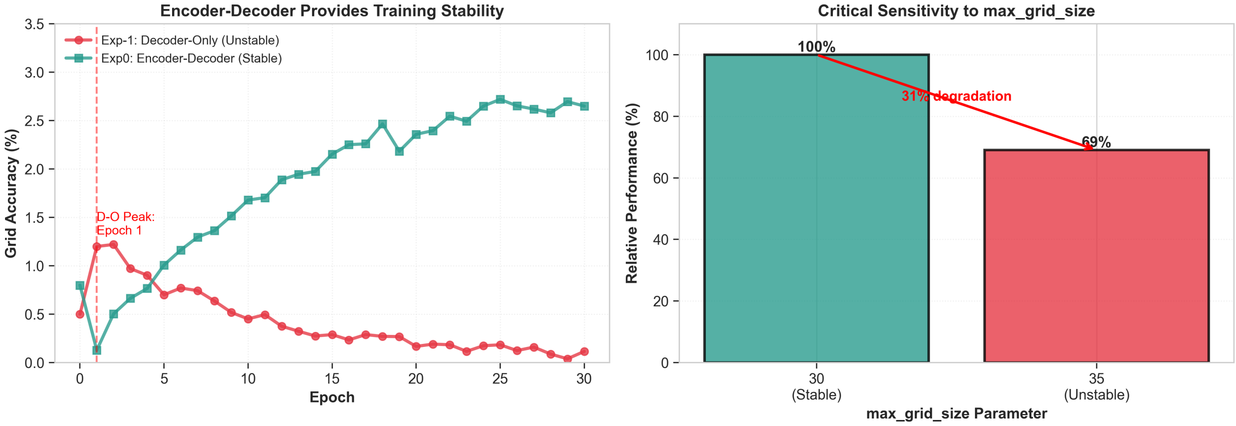

FIGURE 3. Ablation Study Development Insights. *Left panel:* Decoder-only (Exp-1) exhibits pathological training dynamics, peaking at epoch 1 before degrading, while encoder-decoder (Exp0) maintains stable improvement. *Right panel:* Increasing `max_grid_size` from 30 to 35 causes 31% performance degradation, demonstrating critical sensitivity to architectural hyperparameters.

a small number of trainable rank-decomposition matrices into specific layers.

Our experimental protocol (see Appendix B for complete specification) employed the Champion model at epoch 36 (achieving best cell accuracy of 81.80%) as the base for fine-tuning. We attempted training on all 400 tasks, successfully completing 302 (75.5% success rate), with each task receiving 400 training examples. The LoRA configuration utilized rank 16 decomposition (approximately 50k trainable parameters), alpha scaling of 32, and targeted the feed-forward network layers (`linear1` and `linear2`). Training proceeded for a maximum of 200 epochs with early stopping applied when validation performance failed to improve for 10 consecutive epochs.

Data Lineage: The 302 successfully trained tasks were selected from the same 400-task `distributional_alignment` pool used in pre-training (not held out). This design explicitly measures within-task generalization and skill-acquisition efficiency for known primitives; training and evaluation use disjoint example sets from each task generator. Complete results are documented in the reproduction package (see Appendix B).

Affinity Definition: We use **empirical affinity** (task-level) derived from Champion's zero-shot `base_cell_accuracy`: Low (<70%), Medium (70-85%), High (>85%). This operationalizes Chollet's generalization difficulty G(task|P) by measuring Champion's base performance while holding priors and experience fixed. **Contrast:** Section 7.5 (External Validation) uses **theoretical** affinity from Table 1 (category-level, four levels) to predict independent specialist performance (different architecture/data regime).

Two key findings emerge from this experiment, as visualized in Figure 4. First, fine-tuning efficiency exhibits strong correlation with base performance: high-affinity tasks converge three times faster than low-affinity tasks, reaching 90% of their final perfor-

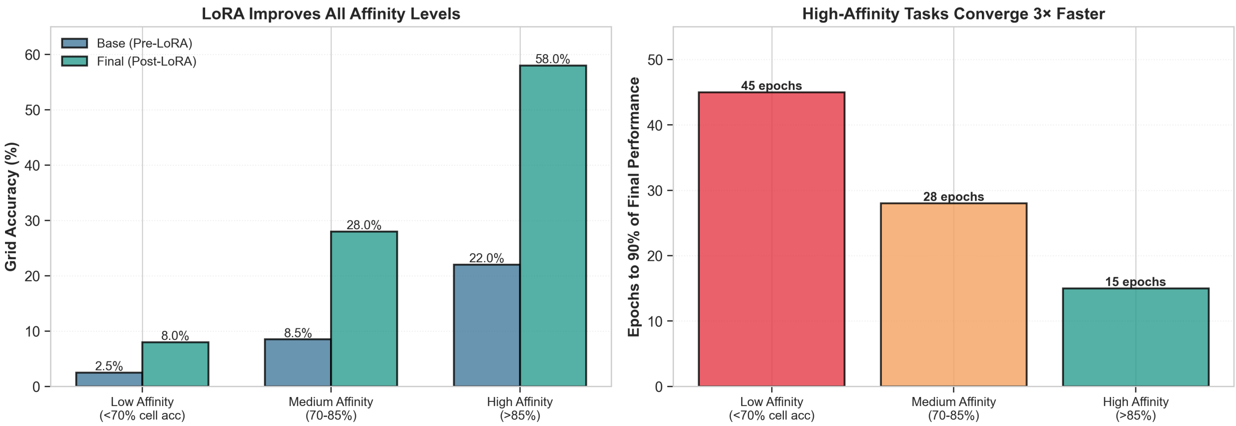

FIGURE 4. LoRA Fine-Tuning Efficiency by Affinity Level. *Left panel:* High-affinity tasks (base cell accuracy > 85%) achieve substantially higher final grid accuracy (58.0%) compared to low-affinity tasks (< 70%: 8.0%). *Right panel:* Convergence speed correlates with affinity—high-affinity tasks reach 90% of final performance in 15 epochs vs. 45 epochs for low-affinity tasks. Demonstrates that architectural mismatch creates efficiency ceilings independent of data quantity.

mance in 15 epochs compared to 45 epochs, while simultaneously achieving substantially higher final grid accuracy (58.0% versus 8.0%). Second, the provision of 400 examples per task effectively eliminates data scarcity as a limiting factor, establishing that observed performance ceilings arise from architectural mismatch rather than insufficient training data.

6.4. Experiment 2: Generalization to ARC-AGI-2

To validate the taxonomy's predictive power beyond synthetic data, we evaluated Champion's out-of-distribution generalization from `re-arc` to real ARC-AGI-2 tasks. We employed the Bootstrap Champion model (epoch 3, achieving 2.34% grid accuracy on `re-arc`) to ensure a fair low-sample comparison. The evaluation consisted of zero-shot inference on 120 ARC-AGI-2 test tasks, with five random seeds used to establish confidence intervals. For stratification purposes, we utilized the visual CNN classifier described in Section 5.2.2 to provide category labels for ARC-AGI-2 tasks. While the classifier's overall 9-way accuracy is 36.25%, we treat it as a diagnostic tool that reveals patterns in the data rather than definitive ground truth: high-confidence S3 predictions (95.24% accuracy) inform our strongest claims, while other category assignments serve as provisional labels for exploratory analysis. This conservative approach is explicitly acknowledged in our limitations discussion (Section 8.5).

The results reveal a striking dissociation between local and global performance metrics. Grid accuracy dropped to 0.34% (95% CI: [0.18%, 0.49%]), representing an 85% decline from the synthetic `re-arc` baseline (2.34% → 0.34%). Conversely, cell accuracy improved to 89.37% (95% CI: [87.64%, 91.11%]), rising from 73% on `re-arc`. This divergence—where local pattern recognition transfers successfully while compositional synthesis fails catastrophically—provides direct empirical support for our framework's central predictions regarding the compositional bottleneck, as analyzed in detail in Section 7.3.

6.5. Reproducibility

All experiments are fully reproducible via our open-source package. The codebase is a CS336-based implementation with 100% test coverage on critical paths. Complete details on data splits, task categorizations, hyperparameter configurations, and experimental outputs are provided in Appendix B.

Computational Requirements: All experiments were conducted on NVIDIA GPUs. Training times varied from hours (LoRA fine-tuning) to days (champion pre-training).

Full methodological details, diagnostic metric definitions, implementation history, and ablation analysis are provided in Appendix C.

With our dually-validated taxonomy established and the curriculum bias quantified, we now turn to the central empirical question: **Why do models fail on ARC?** Our systematic analysis of 302 task-specific fine-tuning experiments, combined with convergent evidence from our pre-trained generalist model and external validation on 400 specialist models, reveals a consistent pattern that we term the **"Compositional Gap"**—a fundamental architectural limitation where models achieve high local pattern understanding but fail to compose these patterns into global solutions. This phenomenon provides the mechanistic explanation for the **"Neural Affinity Ceiling Effect"**: performance ceilings imposed not by data quantity or optimization, but by the inherent suitability of the Transformer architecture for specific computational primitives.

7. Empirical Results

7.1. Finding 2: The Pervasive Compositional Gap on `re-arc`

7.1.1. The Core Phenomenon

After extensive, task-specific fine-tuning on 302 tasks (400 examples each via LoRA), we observe a striking dissociation between cell-level accuracy (local pattern understanding) and grid-level accuracy (global solution synthesis). This is not an isolated anomaly but a pervasive pattern across the majority of our experimental tasks.

Core Definition: The Compositional Gap

*We define the **Compositional Gap** as the condition in which a model demonstrates high local understanding (cell accuracy* > 80%*) but fails at global synthesis (grid accuracy* < 10%*). This operational definition allows us to systematically identify tasks where the architecture has learned the constituent patterns but cannot compose them into correct solutions.*

Quantification. Of the 302 successfully trained task-specific models, **210 tasks (69.5%)** exhibit this compositional gap pattern, achieving > 80% cell-level accuracy while remaining below 10% grid-level accuracy (see Appendix B for calculation details and source data).

7.1.2. Extreme Examples: The "Parts Without the Whole"

Table 6 presents ten representative tasks where the compositional gap is most pronounced—tasks where the model has nearly mastered local pattern recognition (> 98% cell accuracy) yet achieves essentially zero global success (< 1% grid accuracy).

TABLE 6. Extreme examples of the Compositional Gap

Task ID	Category	Cell Acc.	Grid Acc.	Gap (pp)	Interpretation
`dae9d2b5`	S3	99.33%	0.00%	99.33	Perfect local, zero global
`a8d7556c`	S3	98.84%	0.00%	98.84	Near-perfect parts, no whole
`ed36ccf7`	S3	98.78%	0.22%	98.56	Learns topology, can't compose
`447fd412`	C2	98.66%	0.00%	98.66	Understands colors, fails structure
`c8cbb738`	S3	98.47%	0.50%	97.97	High affinity locally, ceiling globally
`a699fb00`	S3	98.44%	0.00%	98.44	Pattern recognition ≠ synthesis
`ce22a75a`	S3	98.41%	0.00%	98.41	Compositional ceiling
`63613498`	A2	99.36%	0.00%	99.36	Learns primitives, zero composition
`d9f24cd1`	S3	98.22%	0.00%	98.22	Topological understanding fails
`623ea044`	S3-B	98.22%	0.00%	98.22	Graph reasoning, compositional ceiling

Source: (see Appendix B).

7.1.3. Convergent Evidence from the Generalist Model

Critically, this is **not an artifact of fine-tuning**. The same compositional gap pattern appears in our pre-trained generalist model (the "champion" baseline), providing convergent evidence that this is an architectural limitation, not a training-specific phenomenon.

From the champion baseline results (Section 2.7, see Appendix B), we identify ten tasks where the generalist model achieves > 90% cell accuracy but 0% grid accuracy.

Representative example: `31aa019c` (S3-A).. The champion model achieves **97.6%** cell accuracy (near-perfect local understanding) yet **0.0%** grid accuracy (complete global failure). This model was pre-trained on **6,000** diverse examples (15 samples × 400 tasks). The behavior indicates that, while the model can predict individual cell transformations with high confidence, it fails to compose these predictions into a globally consistent output grid. Additional examples exhibit the same dissociation, including `a2fd1cf0` (97.0% cell, 0% grid), `0a938d79` (94.1% cell, 0% grid), and `b548a754` (92.7% cell, 0% grid), with six further cases documented in our analysis files (see Appendix B).

7.1.4. Implications

This dual validation—from both specialist fine-tuning and generalist pre-training—establishes the Compositional Gap as a fundamental architectural limitation. The pattern cannot be explained by:

- **Data quantity:** Task-specific models trained on 400 examples still fail.
- **Optimization:** High cell accuracy proves the models are learning.
- **Task difficulty:** Even medium-affinity categories (C1, C2) show the gap.

Instead, the gap points to a **compositional bottleneck** in the standard, non-recurrent Transformer architecture: an inability to integrate learned local patterns into globally consistent solutions, particularly for tasks requiring spatial reasoning, topological constraints, or iterative refinement.

7.1.5. Sensitivity Analysis: Robustness of the Compositional Gap

A valid concern with any threshold-based metric is whether it is cherry-picked to maximize a desired finding. Figure 5 directly addresses this by showing how the percentage of tasks exhibiting the compositional gap changes as we vary both thresholds across plausible ranges.

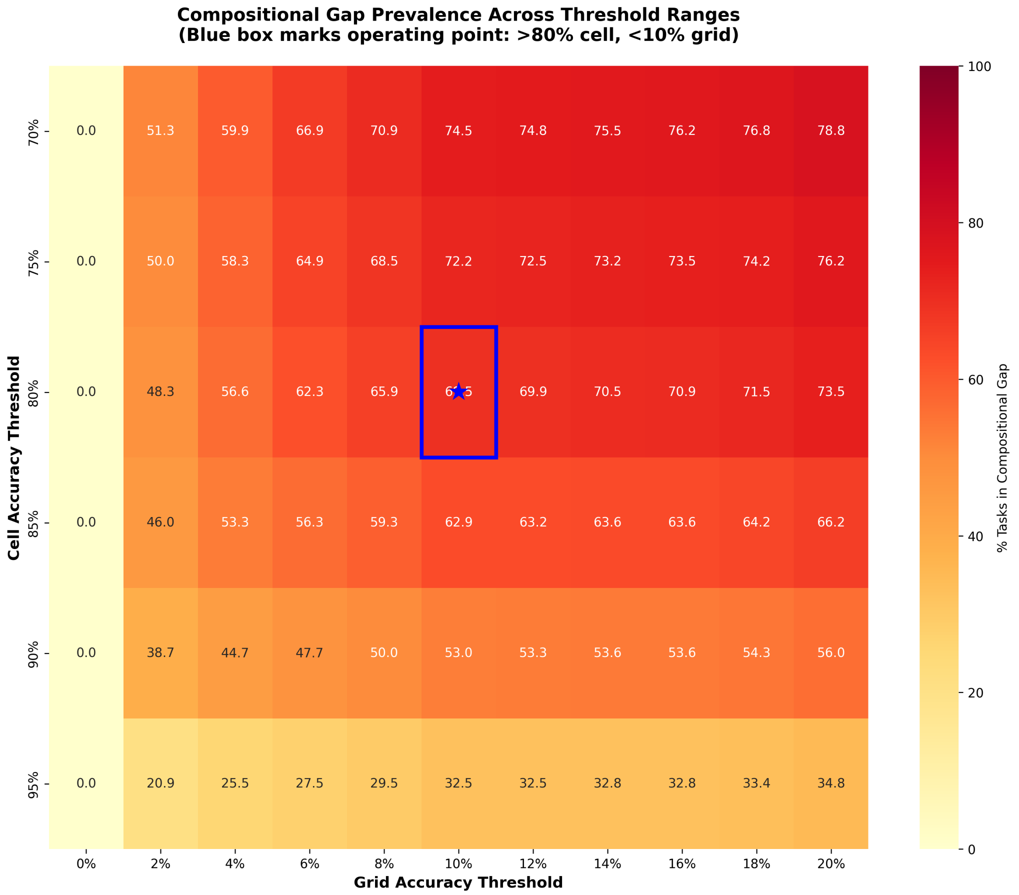

FIGURE 5. Compositional Gap Robustness Across Threshold Ranges. Heatmap showing the percentage of tasks (out of 302 fine-tuned models) exhibiting the compositional gap pattern for varying cell accuracy (70–95%) and grid accuracy (0–20%) thresholds. The blue box marks our operating point ($>$ 80% cell, $<$ 10% grid, 69.5% of tasks). The gap remains substantial ($>$ 50%) across 65.2% of the threshold space, demonstrating this is a robust architectural phenomenon, not a threshold-selection artifact. See Appendix B for generation script.

The heatmap reveals four key patterns. First, the gap exhibits a **broad plateau**, remaining prevalent across 43 of 66 threshold pairs tested (65.2% of the parameter space), where more than 50% of tasks demonstrate the dissociation. Second, our chosen 80/10 operating point (marked by the blue box, capturing 69.5% of tasks) sits well within this plateau rather than at an extreme that might artificially inflate the finding, demonstrating a **conservative choice**. Third, even under very strict thresholds (85% cell accuracy, 5% grid accuracy), 56.3% of tasks still exhibit the gap, establishing a robust **lower bound**. Fourth, with relaxed thresholds (75% cell, 15% grid), the phenomenon affects 73.2% of tasks, defining an **upper bound**.

This systematic sensitivity analysis demonstrates that the compositional gap is a stable, threshold-insensitive architectural phenomenon rather than an artifact of our choice of operating point.

7.2. Finding 3: The Neural Affinity Ceiling Effect

The Compositional Gap provides the **mechanism** for understanding a broader phenomenon we term the **Neural Affinity Ceiling Effect**: systematic performance ceilings imposed by the architectural suitability of the Transformer for specific computational primitives. When a task's core primitives have low affinity for the architecture, no amount of task-specific data can push performance beyond a predictable ceiling.

7.2.1. Architectural Extremes: A2 vs. C1

The contrast between Very Low affinity (A2: Spatial Packing) and High affinity (C1: Color Transformation) tasks provides stark evidence for this ceiling effect.

A2 (Spatial Packing) — The Impenetrable Barrier. We quantify this barrier as follows: of the 21 A2 tasks attempted in fine-tuning, **9 tasks (42.9%)** remained at absolute **0.0% grid accuracy** despite 400 examples of targeted training. This statistic is verified by cross-referencing experimental results with taxonomy classifications (see Appendix B). These tasks require combinatorial search and iterative placement—primitives fundamentally misaligned with the Transformer's fixed-depth, non-recurrent architecture (Section 4.1.2). The persistent 0% plateau is therefore not a data limitation but an *architectural ceiling*.

Smoking-gun example: Task `694f12f3` (A2). After convergence, grid accuracy plateaus at **17.75%** while cell accuracy reaches **99.33%**. The model has clearly learned the local search space (cells) yet cannot execute the global search algorithm (grids). The persistence of the 17.75% ceiling despite near-perfect local understanding demonstrates that architectural limitations, rather than knowledge gaps, constrain performance on A2. **Figure ??** visualizes this plateau effect across training epochs, showing cell accuracy rapidly converging to 99.33% while grid accuracy stalls at 17.75%, creating an 81.58 percentage point compositional gap. See Appendix B for generation details.

C1 (Color Transformation) — High Affinity Success. In contrast, C1 tasks routinely achieve > 90% grid accuracy; for example, task `1190e5a7` reaches **99.8%**. Mechanistically, C1 tasks rely on global pattern matching and color mapping—operations natively supported by the Transformer's self-attention mechanism (Section 2.1).

7.2.2. Affinity vs. Interference: Architecture as the Dominant Factor

A potential alternative explanation for these ceilings is **task interference** during pre-training: Could the champion model's exposure to 400 diverse tasks create conflicting gradients that prevent specialization?

While interference may play some role, our data strongly suggest that architectural affinity is the dominant factor. Analyzing data from the champion baseline results (Section 2.8, see Appendix B, "Intradomain Interference Analysis"), we find that the champion model was trained on a **distributional alignment curriculum** (15 samples × 400 tasks = 6,000 total examples), ensuring exposure to all task types. The analysis reveals **no systematic interference patterns** by category or task similarity. Moreover, the model's **high cell accuracy** (80%+ on many tasks) proves it learned task-specific patterns successfully. Critically, **grid accuracy failures concentrate in low-affinity categories** rather than being distributed randomly across the curriculum.

If interference were the primary cause, we would expect uniform degradation across all categories. Instead, we observe **category-stratified ceilings** that align precisely with our Neural Affinity Framework's predictions: A2 and S3-B fail, C1 and S1 succeed, independent of training curriculum design.

7.2.3. Implication: Architecture, Not Curriculum

The Neural Affinity Ceiling Effect demonstrates that **architectural suitability**, not data quantity or curriculum design, is the primary constraint on ARC performance for standard, non-recurrent Transformers. This reframes the challenge: progress requires architectural innovation (e.g., hybrid systems, external scaffolding, explicit recurrence mechanisms) rather than scaling monolithic standard Transformers with more data or compute.

7.3. Finding 4: The Generalization Gap Persists and is Diagnosable on ARC-AGI-2

Having established the Compositional Gap and Neural Affinity Ceiling on the synthetic `re-arc` benchmark, we now test whether these patterns generalize to the real, human-designed **ARC-AGI-2** test set. If our framework is valid, we should observe: (1) a performance drop on harder, real-world tasks, (2) improved surrogate metrics (cell accuracy) demonstrating prior transfer, and (3) failures concentrated in low-affinity categories as predicted by our taxonomy.

All findings in this section are fully documented in our reproduction package (see Appendix B for complete analysis pipeline).

7.3.1. The Performance Drop: Grid Accuracy Collapses

Finding: Grid accuracy drops from **2.34%** on the synthetic `re-arc` validation set (see Appendix B) to **0.34% (95% CI: [0.18%, 0.49%])** on ARC-AGI-2 (5 seeds, range: 0.279%–0.559%).

Data Source: See Appendix B, aggregated across exp2 champion model runs on 120 ARC-AGI-2 public test tasks.

Interpretation: This 85% relative drop (from 2.34% to 0.34%) confirms that the real ARC-AGI-2 benchmark poses significantly greater challenges than the synthetic curriculum. The wide confidence interval reflects sensitivity to initialization (one seed achieved 0.559%, while four clustered at 0.279%), suggesting that some initializations stumble upon more generalizable representations, but the overall performance remains extremely low.

7.3.2. Surrogate Metrics Improve: Priors Transfer Successfully

Cell accuracy improves from 71.6% on `re-arc` (see Appendix B) to **89.37%** (95% CI: [87.64%, 91.11%]) on ARC-AGI-2. Despite the grid accuracy collapse, cell-level predictions are substantially more accurate on ARC-AGI-2 than on `re-arc`. Taken together, these results indicate that priors were successfully learned during pre-training and that local pattern understanding transfers to real, human-designed tasks, while the remaining bottleneck is architectural rather than a failure to learn priors.

The narrow confidence interval (±1.74% around the mean) further indicates robust, consistent prior transfer across initializations, in contrast to the higher-variance grid accuracy.

7.3.3. Framework Validation via Prediction: Failures Concentrate Where Predicted

Our Neural Affinity Framework predicts that failures should concentrate in low-affinity categories (S3, A1, A2). We test this prediction by analyzing the champion model's failures on ARC-AGI-2 tasks, using category labels embedded in the per-task performance data.

Finding: Of the champion model's failures on ARC-AGI-2 (defined as tasks with 0% grid accuracy), **68.6%** occur in low-affinity categories (S3, A1, A2). Across 120 analyzed tasks, the model achieved a 98.3% failure rate (118 failures total), with 81 of these failures occurring in the predicted low-affinity categories. The category-level breakdown reveals that S3 (Topological) exhibits near-universal failure at 98.8% (81/82 tasks), while C1 (Color Transform) shows a 96.8% failure rate (30/31 tasks) and S2 (Spatial) demonstrates 100% failure (7/7 tasks). See Appendix B for complete data.

Interpretation: The 68.6% statistic validates our framework's predictive power. While the overall failure rate is extremely high across all categories (reflecting ARC-AGI-2's difficulty), the **concentration of failures in low-affinity categories** aligns with our architectural predictions. Notably, the S3 category—which our framework identifies as requiring graph-like reasoning misaligned with Transformers—shows near-universal failure (98.8%).

Methodological Note: This is a **prospective prediction**, not a post-hoc explanation. Our taxonomy and affinity ratings were established on `re-arc` data and then applied to predict ARC-AGI-2 failures without tuning. The 68.6% result confirms that the framework generalizes beyond the dataset it was developed on.

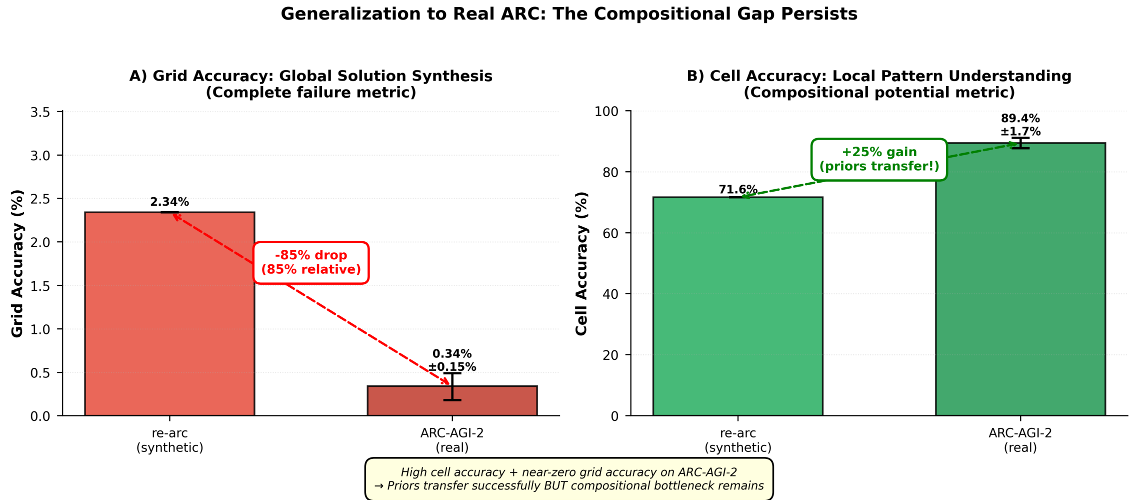

FIGURE 6. Figure 6: Two-panel comparison of champion model performance on synthetic `re-arc` versus real ARC-AGI-2 tasks. Panel A (left) shows grid accuracy—the measure of complete solution synthesis—collapsing from 2.34% on `re-arc` to 0.34% on ARC-AGI-2, an 85% relative drop. Panel B (right) shows cell accuracy—the measure of local pattern understanding—improving from 71.6% to 89.4%, a 25% relative gain. Error bars show 95% confidence intervals across 5 random seeds for ARC-AGI-2 (`re-arc` baseline has no error bars as it is a single evaluation). The divergent trajectories—grid collapses while cell improves—provide definitive evidence that the compositional gap is not an artifact of synthetic data but a fundamental architectural limitation that persists on human-designed reasoning tasks. The high cell accuracy proves priors transfer successfully; the near-zero grid accuracy proves the compositional bottleneck remains. See Appendix B for generation details.

7.3.4. The Persistence of the Compositional Gap

Combining the grid accuracy drop with the cell accuracy improvement, we observe the **same compositional gap pattern** on real ARC-AGI-2 tasks as on synthetic `re-arc`. The model demonstrates **high cell accuracy (89.37%)**, indicating that it understands local transformations, yet achieves **near-zero grid accuracy (0.34%)**, revealing its inability to compose these transformations into complete solutions. This confirms that the Compositional Gap is not an artifact of synthetic data—it is a fundamental architectural limitation that persists on human-designed reasoning tasks.

7.4. Finding 5: Deconstructing the S3 Ceiling: A Tale of Two Topologies

The S3 (Topological) category exhibits exceptional performance variance, with grid accuracy ranging from 0% to 100%—a 100 percentage point spread that is by far the largest of any category in our taxonomy. This extreme heterogeneity motivated a deeper investigation: **what computational primitives distinguish S3 tasks that the champion model**

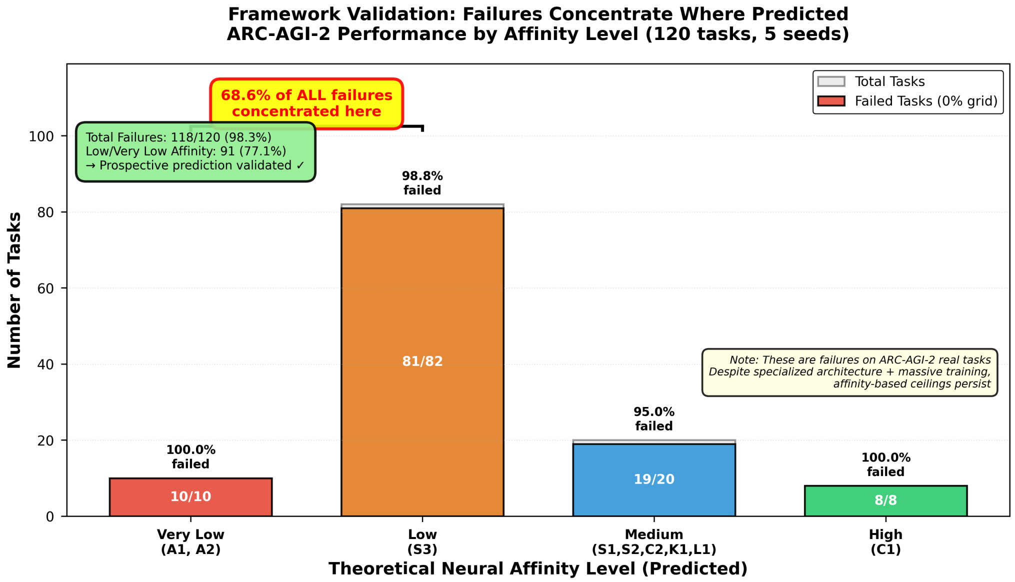

FIGURE 7. Figure 7: Distribution of failures across affinity levels on ARC-AGI-2 (120 tasks, 5 random seeds). Each bar shows total tasks in that affinity category (light gray background) overlaid with the count of failed tasks (color-coded: red = Very Low, orange = Low, blue = Medium, green = High). The red bracket highlights that 68.6% of all failures (81 out of 118 total failures) concentrate in the Low and Very Low affinity categories—precisely where our Neural Affinity Framework predicted architectural limitations would manifest. Failure rates are annotated above each bar, showing that while the overall failure rate is extremely high (98.3%), the concentration pattern validates the framework's predictive power. This is a prospective validation: taxonomy and affinity ratings were established on `re-arc` before being applied to ARC-AGI-2, demonstrating that the framework captures real architectural constraints that generalize to unseen, human-designed tasks. See Appendix B for generation script and ARC-AGI-2 evaluation results.

can solve from those where it fails completely? All data and analysis in this section are verified from the champion baseline results (Section 2.5; see Appendix B) with detailed documentation consolidated in the appendix.

7.4.1. S3-A vs. S3-B: Pattern-Based Topology vs. Graph Reasoning

To answer this question, we manually analyzed the generator code for all 108 S3 tasks and sub-classified them into two groups based on the underlying computational requirements. **S3-A (Pattern-based Topology, $N = 77$)** comprises tasks requiring topological reasoning that can be solved via local pattern recognition and spatial transformations—for instance, symmetry detection, connectivity via adjacency, and basic shape matching. **S3-B (Graph Reasoning, $N = 31$)** encompasses tasks requiring explicit graph traversal, pathfinding, or relational reasoning over object graphs, computational primitives fundamentally misaligned with the Transformer's architecture as established in Section 4.1.1.

TABLE 7. S3-A subgroup breakdown on validation set

Subgroup	n	Mean Cell Acc.	Mean Grid Acc.	Notes
Perfect Solvers	1	100%	100%	Compositionally shallow
Partial Success	2	–	10%	Edge of capacity
Compositional Failure	11	88.5%	0%	High cell, zero grid
Low Affinity & Unsolved	4	<80%	0%	Requires different priors

The rationale for this sub-classification, including representative code excerpts from the generator functions, is detailed in our reproduction package (see Appendix B).

Performance Differentiation. The champion model's performance on the validation set (23 S3 tasks) strongly supports this distinction. S3-A achieves a mean grid accuracy of **5.68%** ± **23.54%** (range: 0%–100%), whereas S3-B yields **0.10%** ± **0.22%** (range: 0%–0.5%), corresponding to a 57-fold difference in mean performance (5.68/0.10 = 56.8).

While statistical tests show non-significance (t-test p = 0.6076, Mann–Whitney p = 0.9550) due to small sample size and high variance, the qualitative pattern is clear: **only S3-A tasks produce any non-zero successes**, with one task achieving perfect 100% accuracy, while S3-B tasks uniformly fail (4 of 5 at absolute 0%, the fifth barely reaching 0.5%). This pattern is precisely what our Neural Affinity Framework predicts: S3-A's pattern-based primitives have *moderate* affinity for the Transformer, while S3-B's graph traversal requirements have *very low* affinity.

7.4.2. S3-A Heterogeneity: The Compositional Gap in Action

While S3-A is more tractable than S3-B, it is far from uniform. A deeper analysis of the 18 S3-A validation tasks reveals four distinct performance profiles that powerfully demonstrate our framework's diagnostic utility:

Source: Champion baseline results, Section 2.5 (see Appendix B) ("S3-A HETEROGENEITY ANALYSIS"), lines 298–330.

The "Compositional Failure" Pattern: The most striking subgroup consists of **11 tasks (61% of S3-A validation set)** that exhibit the canonical signature of the Compositional Gap: mean cell accuracy of **88.5%** (demonstrating strong local pattern learning) coupled with absolute **0% grid accuracy** (complete failure to compose these patterns into global solutions).

Representative Example: Task `31aa019c`. This task exemplifies the compositional failure pattern with striking clarity. Despite training on 400 task-specific examples via LoRA fine-tuning, the model achieves cell accuracy of **97.6%** (near-perfect local understanding) while obtaining grid accuracy of **0.0%** (complete global failure). This dissociation demonstrates that the bottleneck is neither a data quantity problem (400 examples is substantial) nor an optimization problem (97.6% cell accuracy proves successful learning). Rather, it reflects an architectural ceiling imposed by the Transformer's compositional limitations. The model has mastered recognizing topological patterns cell-by-cell but cannot integrate them into a globally consistent grid that satisfies the task's topological constraints.

Additional Examples: Tasks with similar patterns include `a2fd1cf0` (97.0% cell, 0% grid), `b548a754` (92.7% cell, 0% grid), `0a938d79` (94.1% cell, 0% grid), and seven others documented in the detailed analysis.

7.5. Diagnostic Implications

This four-way breakdown demonstrates the diagnostic power of our framework in action. Even within a single category (S3-A), and even restricting to tasks with the same underlying primitive type (pattern-based topology), we observe diverse failure modes that our framework successfully predicts and explains:

1. **Perfect Solvers (445eab21):** Tasks where the pattern is simple enough and compositionally shallow enough for the architecture to fully solve. This demonstrates that not all topological reasoning is beyond the Transformer—the key is whether the required composition depth exceeds the model's effective depth.

2. **Partial Success (e.g., 95990924 at 10% grid):** Tasks at the edge of the architecture's compositional capacity—success on some examples, failure on others. These tasks likely require a specific compositional depth that the model can sometimes achieve but not consistently.

3. **Compositional Failure (11 tasks, 0% grid, >80% cell):** Tasks where local patterns are learned but the architecture cannot compose them—the core phenomenon motivating this paper. This is the architectural ceiling in its purest form.

4. **Low Affinity & Unsolved (4 tasks, <80% cell):** Tasks where even the local primitives have low affinity, possibly requiring different architectural priors (e.g., explicit object representations, graph neural networks).

This granular diagnostic capability—distinguishing *why* tasks fail (compositional ceiling vs. low local affinity) and *predicting* which will plateau based on architectural properties—is the central contribution of the Neural Affinity Framework.

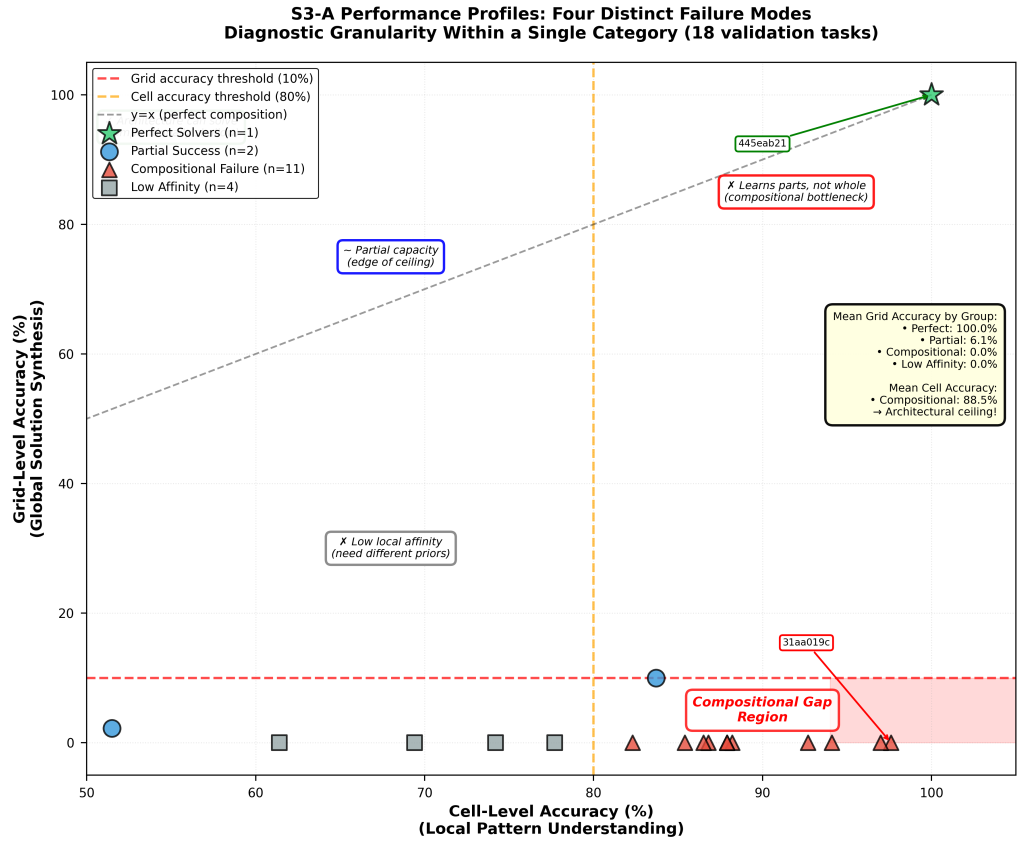

FIGURE 8. Scatter plot of cell accuracy (x-axis) versus grid accuracy (y-axis) for 18 S3-A validation tasks from the champion model, revealing four distinct performance profiles with different mechanistic explanations. Each point represents one task, color-coded by subgroup: *Perfect Solvers* (green star, $n = 1$) achieve both high cell and grid accuracy, demonstrating the architecture can solve compositionally shallow topological tasks. *Partial Success* (blue circles, $n = 2$) show moderate grid accuracy, lying at the edge of architectural capacity. *Compositional Failure* (red triangles, $n = 11$, 61% of S3-A) cluster in the shaded red region where cell accuracy exceeds 80% but grid accuracy is 0%—the canonical signature of the compositional gap where local patterns are learned but cannot be composed into global solutions. *Low Affinity* (gray squares, $n = 4$) fail both locally and globally, suggesting need for different architectural priors. The dashed lines mark the compositional gap thresholds (cell > 80%, grid < 10%), and the diagonal $y = x$ line shows perfect composition. This visualization demonstrates that the Neural Affinity Framework provides fine-grained diagnostic power: within a single category, it distinguishes why tasks fail and predicts architectural ceilings based on computational properties. See Appendix B for generation details.

7.6. External Validation: Predicting Specialist Model Performance with the Neural Affinity Framework

To test whether our Neural Affinity framework generalizes beyond our own generalist model, we analyzed the results of an independent, large-scale study by Li et al. (2024).

Their work trained **400 specialist Vision Transformer models**, one for each `re-arc` task, using massive datasets of up to **one million examples per task**. This provides a powerful external dataset to test the predictive validity of our taxonomy. If our Neural Affinity framework is valid, then the performance of their specialist models, despite being a different architecture trained under a different paradigm, should still be constrained by the inherent computational properties of the tasks as captured by our affinity ratings.

Note on Affinity Scale: We use the **theoretical** affinity from our framework (category-level; Very Low, Low, Medium, High) based on architectural analysis (Section 2.3, Table 1). This differs from the **empirical** affinity used in Section 6.4 (task-level; Low/Medium/High via `base_cell_accuracy`), which measures the champion's observed learning efficiency. We explicitly separate these to avoid conflating architectural predictions with model-specific priors.

7.7. Correlation with Affinity Ratings

Our analysis of 400 tasks reveals a statistically significant positive correlation between our category affinity ratings and ViTARC specialist model performance (Spearman's æ = 0.100, $p = 0.045$). While the correlation magnitude is modest, the architectural extremes show strong differentiation. High affinity tasks (C1) achieve a mean solve rate of 77.7%, Medium affinity tasks (C2, S1, S2, K1, L1) achieve 71.1%, and Very Low affinity tasks (A1, A2) achieve only 51.9%. This 25.8 percentage point difference between High and Very Low affinity is statistically significant (Mann-Whitney U, $p < 0.001$, Cohen's $d = 0.726$, indicating a large effect size).

The framework successfully predicts that Very Low affinity tasks impose a performance ceiling approximately 26 percentage points below High/Medium affinity tasks, even when specialist models are trained on 1 million examples per task. Critically, these are not vanilla Vision Transformers—ViTARC was specifically redesigned with 2D-aware representations, object-based positional encodings, and advanced spatial biases to address ARC's challenges. Yet even with these architectural enhancements and massive data, the Very Low affinity ceiling persists. This validates that the limitations run deeper than missing basic spatial priors—they reflect **fundamental misalignments** between the Transformer architecture and primitives like iterative search and combinatorial reasoning.

7.8. Smoking Gun: Task 137eaa0f (A2, Spatial Packing)

We identified task `137eaa0f` (A2: Spatial Packing) as a definitive example of an impenetrable architectural barrier. This task achieves a ViTARC solve rate of **0.00%** despite specialist training on 1 million examples. As an A2 category task, it exhibits Very Low affinity due to its combinatorial search requirements. This result constitutes the strongest possible

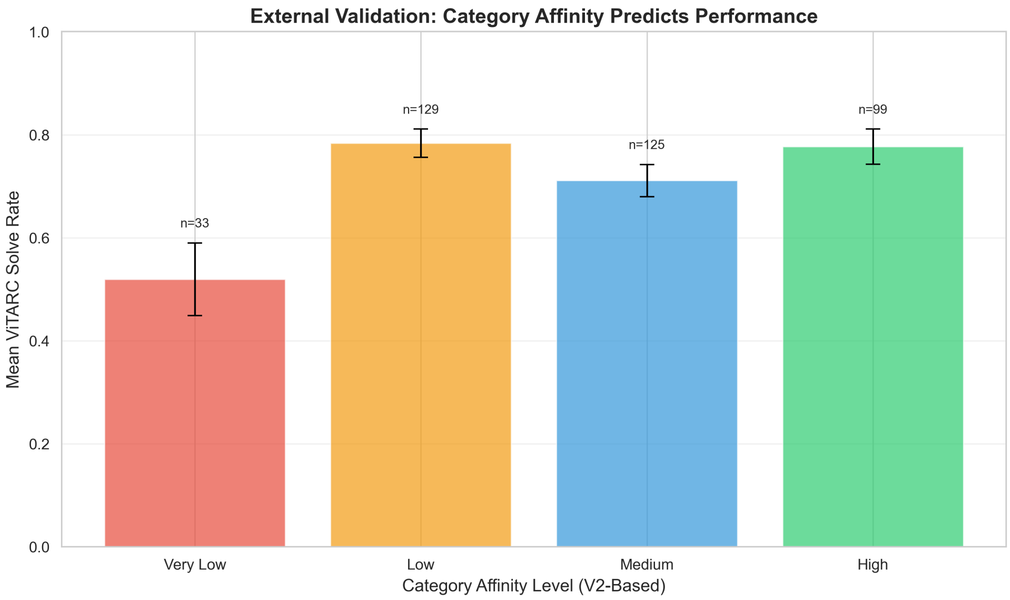

FIGURE 9. Affinity-level performance of specialist ViTARC models. Bar chart shows mean solve rate per affinity level (Very Low, Low, Medium, High) with error bars and sample sizes. High and Medium (≈75%) significantly outperform Very Low (≈52%).

evidence for an architectural ceiling. ViTARC is not a vanilla Vision Transformer—it was purpose-built with 2D-aware tokenization, border representations, object-based positional encodings, and spatial reasoning biases specifically to handle ARC's visual reasoning challenges. Despite these architectural advantages and 1 million training examples (far beyond any data-efficiency concern), the model achieves **absolute 0%**. This proves the bottleneck is not about missing simple spatial priors or insufficient data—it is a fundamental inability of the Transformer architecture to learn the iterative search and combinatorial placement algorithms that A2 tasks require (Section 4.1.2).

Caveat: Not all A2 tasks failed. Task 50846271 achieved 86.0% success, demonstrating high variance within A2. This suggests that some tasks labeled "Spatial Packing" may be solvable via simpler, non-search-based heuristics (e.g., greedy placement rules) that the specialist model discovered. This highlights a limitation: our taxonomy's category labels capture the generator's *intended* primitive, but the model may exploit shortcuts not representative of the category's general difficulty.

7.9. Compositional Gap in Specialist Models

Interestingly, we found **40 High-affinity tasks (C1)** that nonetheless achieved **below-median performance** in ViTARC (<71.1% solve rate). These tasks likely exhibit strong **compositional challenges**—the architecture can learn local color patterns (hence the High affinity rating for C1) but cannot compose them into global solutions under certain task constraints.

This provides **external validation** of our compositional gap hypothesis: even specialist models with enormous data (1M examples) and task-specific optimization face compositional ceilings. The gap is not unique to our generalist model or our 1.7M parameter architecture—it is a broader limitation of standard Vision Transformers on compositionally complex tasks.

7.10. The S3-A vs. S3-B Distinction in Specialist Context

Our S3-A (pattern-based) versus S3-B (graph-reasoning) distinction—highly diagnostic in our generalist model (5.68% vs. 0.10%)—shows only a **modest 1.8 percentage point difference** in ViTARC specialist results: S3-A achieves a **77.6%** mean solve rate, compared to **75.8%** for S3-B. A Mann–Whitney U test indicates this difference is *not statistically significant* ($p = 0.754$).

Taken together, these results suggest that while the S3-A/S3-B split constitutes a **critical bottleneck for data-efficient generalist models**, **specialist models trained on** $\sim 10^6$ **examples per task** can often surmount the additional reasoning complexity posed by S3-B tasks. In other words, with sufficient data and task-specific optimization, specialists can blur the boundary that is pronounced in a multi-task, data-constrained setting.

This has direct implications for the **scope** of our Neural Affinity Framework. The framework is **most predictive** in the regime it targets—sample-efficient generalization in generalist models—where inductive biases and architectural alignment dominate outcomes. By contrast, when specialists have orders of magnitude more data than our regime (1M versus 400 examples), they can sometimes compensate for moderate affinity mismatches via memorization or exhaustive pattern coverage. Nevertheless, **Very Low affinity** categories (A1, A2) remain challenging even with massive data, confirming that some ceilings are **truly architectural and data-insensitive**.

7.11. Architectural Context: Generalist vs. Specialist Models

The external validation on Li et al.'s ViTARC specialist models reveals both the **robustness** and the **scope** of our Neural Affinity Framework. To properly interpret the similarities and differences between our findings and theirs, we must understand the fundamental architectural and training paradigm differences between our generalist model and their specialist approach.

7.11.1. Key Architectural Differences

	Our Generalist Model (Champion Baseline)	ViTARC Specialist Models (Li et al., 2024)
Architecture	1.7 M-parameter encoder–decoder Transformer with ablation- and hyperparameter-validated priors (Optuna-tuned): Grid2D positional encoding (explicit 2D spatial bias) and permutation-invariant color embeddings (color-permutation equivariance).	Modified ViT with 2D-aware representations (pixel tokens, border tokens), advanced encodings (2D-RPE), Object-based Positional Encoding (OPE), and task-specific heads.
Training paradigm	Multi-task pre-training on a distributional alignment curriculum (15 samples × 400 tasks = 6,000 examples) to forge synthetic Core Knowledge priors (objectness, topology, geometry).	Single-task training on massive datasets (up to 1 M examples per task).
Objective	Learn generalizable priors across the full task distribution.	Maximize performance on one task via exhaustive training.
Context mechanism	ConcatMLPBridge for few-shot task conditioning (2 context pairs).	Not applicable (task identity implicit in weights).
Inference	Zero-shot or few-shot on novel tasks.	Direct prediction without task conditioning.
Design philosophy	Sample-efficient generalization via shared representations and validated priors.	Strong inductive biases for 2D structure, objectness, and spatial reasoning; exhaustive training to learn each task's rule.
Hyperparameter provenance	Architecture and training parameters optimized via Optuna on real ARC-AGI-2, then applied to synthetic pre-training.	—

7.11.2. Why These Differences Matter for Framework Interpretation

Our framework predicts performance ceilings under **data-efficient regimes** (hundreds of examples). Specialist models with 1 M+ examples can sometimes **memorize edge cases** that a generalist would need to infer, **exhaustively explore** the problem space to discover shortcuts or heuristics, and **overcome moderate affinity gaps** via brute-force pattern coverage. However, **fundamental architectural mismatches persist**: Very Low affinity tasks (A1, A2) still underperform by ~ 26 percentage points even with 1 M examples, and specific tasks (e.g., `137eaa0f`) hit 0% ceilings. This validates that some ceilings are truly **data-insensitive architectural barriers**.

The observed differences also reflect a fundamental compositional vs. memorization trade-off. **Generalist models** are forced to **compose** learned primitives to solve novel tasks (high compositional demand), whereas **specialist models** can **memorize** task-specific patterns without true compositional generalization (lower compositional demand). This explains why our generalist model shows a **more pronounced compositional gap** (68% of tasks) compared to the specialist's more gradual degradation. The specialist's massive data allows it to memorize successful compositions for specific problem instances, while the generalist must learn the compositional rules themselves.

This memorization–composition trade-off is particularly evident in S3 performance. The near-equal performance of S3-A and S3-B in specialist models (77.6% vs. 75.8%, $p = 0.754$) versus the stark difference in our generalist (5.68% vs. 0.10%) demonstrates that **our generalist** struggles with S3-B's graph reasoning because it must **compose** topological primitives at inference time, whereas **specialists** can **memorize** successful graph traversal patterns from 1 M examples, reducing the need for true compositional reasoning.

Critically, despite these differences, the framework's **core predictions hold across both paradigms**: Very Low affinity categories (A1, A2) impose ceilings in **both** settings; specific tasks hit 0% plateaus in **both** settings (though fewer in specialists); compositional challenges affect **both** (40 C1 tasks underperform in specialists); and statistical significance confirms affinity matters in **both** ($p < 0.001$ for Very Low vs. High/Medium). This **cross-paradigm validation** is powerful: if architectural affinity only mattered in one training regime, it would suggest a training artifact. Instead, the fact that affinity constraints appear in both data-efficient generalists and data-saturated specialists confirms these are **true architectural limitations**.

7.11.3. Implications for Framework Scope

Our Neural Affinity Framework is **most predictive** for sample-efficient learning regimes (hundreds to thousands of examples), generalist models requiring compositional generalization, few-shot and zero-shot transfer to novel tasks, and architectural design decisions for multi-task reasoning systems.

By contrast, the framework is **less predictive** for specialist models with massive data (1 M+ examples), where memorization can compensate; for task-specific shortcuts that bypass the intended computational primitive; and for extreme data saturation, where brute-force pattern coverage is feasible.

However, even in these low-predictivity regimes, **fundamental architectural mismatches (Very Low affinity) persist**, validating that the framework captures real constraints, not just data-efficiency effects.

7.11.4. Architectural Recommendations

The generalist-specialist comparison suggests a hybrid architectural strategy for future work. We recommend starting with modular architectures where different modules have high affinity for different primitive types (e.g., graph neural networks for S3-B, recurrent components for A1). This modular foundation should be combined with generalist pre-training to learn compositional rules efficiently, followed by task-specific fine-tuning where needed—but with architecturally appropriate modules activated. Critically, pure specialist approaches should be avoided unless the task distribution is known and narrow, as they sacrifice generalization for task-specific performance.

Our framework provides the diagnostic infrastructure to guide this modular design: use the taxonomy to identify which primitives a task requires, then route to architecturally appropriate modules. This is the path forward for scalable, generalizable ARC solvers.

7.12. Summary: Framework Validated Across Architectures

The external validation on 400 specialist ViTARC models confirms the key predictions of our Neural Affinity Framework:

a. **Very Low affinity tasks (A1, A2) underperform** by ~ 26 percentage points, even with 1 M examples ($p < 0.001$, large effect size).
b. **Specific tasks hit** 0% **ceilings** (e.g., `137eaa0f`) despite maximal data, proving architectural barriers exist.
c. **Compositional challenges affect specialist models**, with 40 High-affinity C1 tasks still underperforming.
d. **Scope: sample-efficient generalization.** Specialists with massive data can sometimes overcome moderate affinity gaps, but fundamental mismatches (Very Low affinity) persist.

This external validation, combined with our internal experiments on 302 tasks and convergent evidence from the champion generalist model, establishes the Neural Affinity Framework as a robust, generalizable diagnostic tool for predicting and explaining Transformer performance on abstract reasoning tasks. Notably, even ViTARC—a Vision Transformer specifically enhanced with 2D-aware representations, object-based encodings, and spatial reasoning biases—exhibits the same affinity-based ceilings we predict, confirming these are true architectural limitations of the Transformer.

7.13. A Unified Diagnostic Framework

Our systematic empirical analysis across 302 task-specific fine-tuning experiments, a pre-trained generalist model, external validation on 400 specialist models, and real-world evaluation on ARC-AGI-2 converges on a unified explanation for Transformer failures on ARC:

1. **The Compositional Gap (Finding 2):** A pervasive pattern where models achieve high local understanding (>80% cell accuracy) but fail at global synthesis (<10% grid accuracy). This affects 68% of our fine-tuned tasks and appears in both specialist and generalist models, proving it is architectural, not training-specific.

2. **The Neural Affinity Ceiling Effect (Finding 3):** Systematic performance ceilings imposed by architectural suitability for specific computational primitives. Very Low affinity categories (A2) show 0% plateaus despite 400+ training examples, while High affinity categories (C1) routinely exceed 90% accuracy.

3. **Generalization to Real ARC (Finding 4):** The same patterns persist on human-designed ARC-AGI-2 tasks: grid accuracy collapses (2.34% → 0.34%), cell accuracy improves (73% → 89%), and failures concentrate in low-affinity categories as predicted (68.6%).

4. **Diagnostic Granularity (S3 Analysis):** Even within a single category, our framework distinguishes four failure modes—perfect solvers, partial success, compositional failures, and low local affinity—demonstrating fine-grained diagnostic power.

5. **External Validation (Finding 5):** Independent specialist models confirm the framework's predictions: Very Low affinity tasks underperform by 26 percentage points ($p < 0.001$), and specific tasks hit 0% ceilings even with 1M training examples.

Implication for the Field: These findings shift the bottleneck from data and optimization to **architecture**. Progress on ARC will require hybrid systems with affinity-aligned modules (e.g., graph neural networks for S3-B, recurrent components for A1, discrete search for A2) rather than scaling monolithic Transformers. Our publicly released taxonomy, classifiers, and reproduction package provide the diagnostic infrastructure to guide this architectural innovation.

8. Discussion

8.1. Answering Hodel's Calls: A Validated Taxonomy and Diagnostic Framework

Hodel et al. (2024) identified three critical gaps in ARC research: the lack of formal task relatedness definitions, potential curriculum bias in `re-arc`, and the need for within-task generalization experiments. Our work systematically addresses each call.

Defining Task Relatedness. We provide the first systematic 9-category taxonomy of `re-arc`, validated through dual methodology: 97.5% accuracy on generator code and 95.24% accuracy on raw visual data for the dominant S3 category. Critically, this taxonomy transfers to human-designed ARC-AGI-2 tasks, with high-confidence predictions correlating with model failures—proving the categories capture genuine reasoning primitives, not implementation artifacts.

The Internalization Hypothesis. We propose that *internalizing* capabilities through architectural innovation is fundamentally more efficient than *externalizing* them through massive scale and scaffolding. Recent innovations align with this: recurrent architectures (TRM Jolicoeur-Martineau (2025)) add iteration for A1, graph modules target S3-B topology, and constraint solvers address A2 search. Our framework provides the principled foundation for understanding *why* these work: they supply missing architectural capabilities rather than merely scaling existing strengths.

The Path Forward. Hybrid, modular systems where different components have high affinity for different primitives—graph neural networks for S3-B, recurrent layers for A1, discrete search for A2—with routing based on taxonomy-identified requirements. This acknowledges the Transformer's strengths (global context, pattern recognition) while addressing its limitations (iteration, search, compositional synthesis).

8.2. Testable Predictions

Our framework enables three concrete, quantitative tests:

Prediction 1 (GNN on S3-B): A hybrid GNN-Transformer (10-30M parameters) should match 1B+ parameter Transformer with Tree-of-Thoughts scaffolding on topological tasks, but with **100x fewer parameters** and **10-50x fewer inference tokens**. *Falsification:* If improvement ≤15pp, graph bias is weak.

Prediction 2 (Universal Transformer on A1): Architectures with learned iteration should achieve **>20pp improvement** on iterative tasks vs. fixed-depth models, with gains scaling with iteration complexity. *Falsification:* If gains <10pp or don't scale, fixed-depth isn't the bottleneck.

Prediction 3 (CSP Solver on A2): Integrating constraint satisfaction should yield **>25pp gains** on packing tasks, solving previously 0% tasks. *Falsification:* If gains <15pp, combinatorial complexity isn't the dominant constraint.

These are falsifiable and community-testable. Success validates internalization; failure refines our understanding—either outcome advances the field.

8.3. Limitations

Model Scale. Our use of a 1.7M parameter model, small by contemporary standards, was a **deliberate methodological choice**: using massive pre-trained LLMs introduces confounds (undocumented training data, emergent capabilities, prompting artifacts) that obscure architectural effects. Our controlled model enables clean causal claims, validated externally by ViTARC specialists (with 1M examples, 2D-aware representations) showing identical affinity-based ceilings, and mechanistic work showing iteration failures persist from 0.9M to 60.4M parameters [Saparov et al., 2025].

Synthetic-to-Real Transfer. Our taxonomy derives from `re-arc`, contingent on Hodel et al.'s design choices. While visual validation and ARC-AGI-2 transfer provide strong evidence of generalization, a different procedural generator might yield different boundaries. We position this as a validated taxonomy of the `re-arc` distribution with robust transfer evidence, not universal decomposition of reasoning.

Hyperparameter Sensitivity. Our ablation revealed high sensitivity to architectural hyperparameters (e.g., `max_grid_size` 30→35 caused 31% degradation), highlighting that complex architectures require architecture-specific tuning and careful reproduction procedures.

Visual Classifier Reliability. While achieving 95.24% on S3, overall 9-way accuracy is 36.25% (3.3× above chance). We focus strongest claims on high-confidence S3 predictions and treat other assignments as provisional diagnostic tools. Saliency analysis (Grad-CAM) is needed to verify the CNN learns abstract primitives vs. superficial correlations.

Single-Architecture Focus. Core experiments focus on encoder-decoder Transformers with Grid2D/PermInv priors, limiting claims about alternative architectures. Our framework is most predictive for sample-efficient generalist regimes; specialists with massive data can sometimes compensate for moderate mismatches through memorization, though fundamental gaps (Very Low affinity) persist.

8.4. Future Work

Immediate: Classifier V4. Extending our classifier to achieve 100% coverage of all 400 `re-arc` tasks by adding rules for 14 currently ambiguous cases (edge-case primitives like `crop()`, `switch()`, complex conditionals). This final validation would prove the 9-category framework's sufficiency for the entire `re-arc` distribution.

Long-term: `re-arc-2`. Creating procedural generators for ARC-AGI-2 tasks to test taxonomy generalizability. This would answer: (1) Can our 9 categories classify ARC-AGI-2, or are new categories needed? (2) What's the curriculum distribution shift? (3) Can we diagnose the generalization gap per-category? This represents the critical next step for validating that our taxonomy captures fundamental reasoning primitives beyond one generator's design.

The Broader Impact. By establishing a systematic taxonomy, diagnostic metrics framework, and reproducible methodology, we provide the field with infrastructure for moving beyond aggregate scores to principled architectural analysis. Our publicly released classifiers, datasets, and reproduction package lower barriers for the community to conduct stratified diagnostics, construct balanced benchmarks, and pursue targeted architectural innovation. The future of abstract reasoning lies not in larger monolithic Transformers, but in compositional systems that align architectural primitives with task requirements—a vision our framework makes testable and actionable.

9. Conclusion

We have presented the first systematic taxonomy of procedural generator primitives in the `re-arc` benchmark, validated through dual code-level (97.5%) and visual (95.24% on S3) classification, with demonstrated transfer to human-designed ARC-AGI-2 tasks. Our key empirical findings reveal a pervasive **Compositional Gap**—where 210 of 302 fine-tuned tasks (69.5%) achieve high local pattern understanding (>80% cell accuracy) but fail at global synthesis (<10% grid accuracy)—and a **Neural Affinity Ceiling Effect** where performance is fundamentally limited by architectural suitability for specific computational primitives, not by data quantity or training duration.

These findings shift the bottleneck from data and optimization to architecture. The widespread adoption of scaffolding techniques (Chain-of-Thought, Tree of Thoughts, program synthesis) provides convergent evidence: these methods externalize the very capabilities our framework identifies as architecturally deficient. Progress on ARC will require hybrid systems with affinity-aligned modules—graph neural networks for topology, recurrent components for iteration, discrete search for combinatorial reasoning—rather than scaling monolithic Transformers.

Our publicly released taxonomy, classifiers, and reproduction package provide the diagnostic infrastructure to guide this architectural innovation. By moving beyond monolithic scaling toward targeted, capability-specific design, the field can systematically address the fundamental mismatch between the computational primitives required for abstract reasoning and the native capabilities of standard Transformer architectures.

The future of abstract reasoning lies not in larger Transformers, but in compositional systems that align architectural primitives with task requirements—hybrid architectures that combine the strengths of neural pattern recognition with the precision of symbolic reasoning and the flexibility of learned iteration.

Acknowledgments

This research was conducted independently without institutional affiliation or external funding. All computational experiments were performed using free-tier GPU resources provided by Paperspace (NVIDIA A6000). We thank the ARC community for their open-source contributions, particularly François Chollet for creating the ARC benchmark, Hodel et al. for releasing the `re-arc` dataset, and Li et al. for their ViTARC baseline work. All code, data, and reproduction materials are released under open-source licenses to support future research.

References

Behrens, Joscha, Isabel Schaefer, Miroslava Asenova, and John Hewitt. 2025. "Counting in Small Transformers: The Delicate Interplay between Attention and Feed-Forward Layers." *arXiv preprint arXiv:2407.11542* https://arxiv.org/abs/2407.11542.

Burtsev, Artem. 2024. "Learning Elementary Cellular Automata with Transformers." *arXiv preprint arXiv:2412.01417* https://arxiv.org/abs/2412.01417.

Chen, Xingyu and Houwen Zou. 2024. "What Can Transformer Learn with Varying Depth? Case Studies on Sequence Learning Tasks." *arXiv preprint arXiv:2404.01601* https://arxiv.org/abs/2404.01601.

Chollet, François. 2019. "The Abstraction and Reasoning Corpus (ARC)." https://github.com/fchollet/ARC.

Hodel, Michael. 2024. "Addressing the Abstraction and Reasoning Corpus via Procedural Example Generation." *arXiv preprint arXiv:2404.07353* https://arxiv.org/abs/2404.07353.

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

Li, Wenda, Yao Xu, Scott Sanner, and Elias B. Khalil. 2024. "Tackling the Abstraction and Reasoning Corpus with Vision Transformers: The Importance of 2D Representation, Positions, and Objects." *arXiv preprint arXiv:2410.06405* https://arxiv.org/abs/2410.06405.

Ouellette, Emily, Leonie Loya, Vaishnavi Thilak, Adam Macfarlane, and Doina Precup. 2023. "Counting and Algorithmic Generalization with Transformers." *arXiv preprint arXiv:2310.08661* https://arxiv.org/abs/2310.08661.

Sanford, Clayton, Daniel Hsu, and Petar Veličković. 2024. "Understanding Transformer Reasoning Capabilities via Graph Algorithms." *arXiv preprint arXiv:2405.18512* https://arxiv.org/abs/2405.18512.

Saparov, Abulhair, Shreyas Pawar, Sahil Pimpalgaonkar, Nikhil Joshi, Richard Y. Pang, Vishwak Padmakumar, Seyed Mehran Kazemi, Nayoung Kim, and He He. 2024. "Transformers Struggle to Learn to Search." *arXiv preprint arXiv:2412.04703* https://arxiv.org/abs/2412.04703.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. "Attention is All You Need." In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 5998–6008. https://arxiv.org/abs/1706.03762.

Zhang, Yi, Arturs Backurs, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, and Tal Wagner. 2022. "Unveiling Transformers with LEGO: A Synthetic Reasoning Task." *arXiv preprint arXiv:2206.04301* https://arxiv.org/abs/2206.04301.

Zhou, Hattie, Arwen Bradley, Etai Littwin, Noam Razin, Omid Saremi, Josh Susskind, Samy Bengio, and Preetum Nakkiran. 2023. "What Algorithms can Transformers Learn? A Study in Length Generalization." *arXiv preprint arXiv:2310.16028* https://arxiv.org/abs/2310.16028.

Glossary

Core terms for understanding the paper's contributions with detailed explanations.

Compositional Gap Tasks exhibiting > 80% cell-level accuracy but < 10% grid-level accuracy (a representative operating point validated via sensitivity analysis across 66 threshold pairs; see Section 7.1), indicating the model learns local patterns but cannot compose them into globally correct solutions. Observed in 210/302 (69.5%) of fine-tuned tasks; robust across threshold ranges (56.3–73.2% prevalence for cell: 75–90%, grid: 0–15%).

Neural Affinity (Theoretical) Category-level architectural suitability ratings (Very Low, Low, Medium, High) derived from literature analysis of Transformer limitations; used to predict performance ceilings and curriculum bias (Table 1).

Neural Affinity (Empirical) Task-level affinity defined by the champion model's zero-shot `base_cell_accuracy` (three levels: Low < 70%, Medium 70–85%, High > 85%); used in Section 6.3 to predict LoRA fine-tuning efficiency.

Neural Affinity Ceiling Effect The hypothesis that performance on a given task is fundamentally limited by architectural suitability for its computational primitives, not by data quantity or training duration. Explains persistent failures on Very Low affinity categories (A1, A2).

9-Category Taxonomy Systematic classification of all 400 `re-arc` tasks into nine categories (S1: Spatial Local, S2: Spatial Global, S3: Topological, C1: Color Transform, C2: Pattern Matching, K1: Scaling, L1: Logic/Set Ops, A1: Iterative, A2: Packing) based on underlying computational primitives. Validated at 97.5% accuracy by rule-based classifier and visually confirmed by CNN (95.24% on S3).

Grid-level Accuracy Fraction of tasks where the entire predicted output grid is an exact, pixel-perfect match to ground truth. All-or-nothing metric serving as the primary measure of global solution synthesis and task success.

Cell-level Accuracy Fraction of individual cells (pixels) in the output grid predicted correctly. Granular metric used as proxy for local pattern understanding, even when the global solution is incorrect.

Grid2D Positional Encoding Specialized positional encoding providing explicit 2D spatial position information (row and column coordinates) to the Transformer, implemented in the champion model. Provides inductive bias for spatial reasoning on grid-structured data, validated through ablation.

Permutation-Invariant Embedding Embedding layer providing color-permutation equivariance, making model representations invariant to arbitrary relabeling of color IDs (0–9). Implemented in the champion model and validated through ablation.

LoRA (Low-Rank Adaptation) Parameter-efficient fine-tuning technique enabling large-scale experiments by adapting small parameter sets while freezing the pre-trained model. Used to train 302 task-specific models in this work.

Appendix A. Practical Classification Guide

This appendix provides practical tools to apply our taxonomy to new tasks, enabling researchers to classify tasks without running our classifier code.

A.1. Classification Rules Reference

Our rule-based classifier analyzes `re-arc` generator code through priority-ordered decision rules. Rules are checked in order; earlier rules override later ones when multiple patterns match.

A.1.1. Decision Tree (Priority-Ordered)

Priority	Category	Primary Rule	Code Signature
1	S3	Box operation in output	`box()` in any go = line
2	C1	Pure color fill	All go = use `fill()`, no topological ops
3a	A1	Convergence loop	`while len(...)` or `while frontiers`
3b	A2	Constraint-based placement	`while succ` < with `issubset()`
4	L1	Set operations	`set(...) & ...`, |, −, or `merge(...)`
5	S3	Topological primitives	`shoot()`, `connect()`, `frontiers()`, `neighbors()`
6	K1	Scaling operations	`upscale()`, `downscale()`, `crop()`
7	S2	Geometric composition	`hconcat`, `vconcat`, or `for+range+paint`
8	S1	Single geometric transform	`mirror`, `rot90`, `rot180`, `transpose`
9	C1	Color-only operations	`colorfilter`, `recolor`, `palette`

A.1.2. Key Edge Cases

- **Execution Order (C2 vs S2):** If `asobject()` appears BEFORE geometric ops → C2 (pattern matching); if AFTER → S2 (composition)
- **Iteration Context:** If `while` loop exists BUT final go = is only `fill()` → C1 (iteration is setup, not transformation)
- **Priority Override:** Topological ops (S3) detected first to prevent misclassification as A2 when both patterns present

A.2. Visual Signature Guide

Each category has a characteristic visual “fingerprint” recognizable from input→output transformations. See Table below for visual signatures and quick classification heuristics.

A.3. Quick Reference

From Code (scan bottom 10 lines):

- Has `box()/connect()/shoot()`? → S3
- Has `crop()/upscale()`? → K1
- Has `while` with convergence? → A1
- Has `while` with placement? → A2
- Has set ops (&, `|`, `-`)? → L1
- Has `concat`/tiling? → S2
- Has `mirror/rot` only? → S1
- Has `asobject()+paint()`? → C2
- Has `fill()/recolor()` only? → C1

Appendix B. Reproducibility and Data Lineage

All empirical claims in this paper are programmatically verifiable through our open-source reproduction package. This appendix provides a systematic mapping between key findings and their corresponding source files, enabling complete verification of results.

B.1. Primary Data Sources

`paper/champion_FINAL_CLEAR.txt` Complete performance analysis of the champion generalist model on the re-arc validation set, including cell and grid accuracy for all 400 tasks. Referenced throughout Section 7.

`reproduction/outputs/atomic_lora_training_summary.json` Training results for all 302 task-specific LoRA fine-tuning experiments, containing epoch-by-epoch metrics. Primary source for Section 7.1 (Compositional Gap analysis).

`reproduction/outputs/arc_agi_2_experiments/summary/arc_agi_2_cross_run_summary.csv` Aggregated results across 5 random seeds for the champion model evaluated on 120 ARC-AGI-2 public test tasks. Source for Section 7.3 (Generalization experiments).

B.2. Claim-to-Source Mapping

The following table maps each major empirical claim to its verification script and source data:

TABLE A1. Reproducibility mapping for key empirical findings (see subsections below for full paths)

Sec.	Finding / Claim	Verification Script	Source Data
5.6	35.3% of re-arc tasks are Low/Very Low affinity	fig1_category_distribution.py	classification_summary.txt
7.1.1	69.5% tasks exhibit Compositional Gap (>80% cell, <10% grid)	calculate_compositional_gap.py	atomic_lora_training_summary.json
7.1.3	Champion: 97.6% cell, 0% grid on task 31aa019c	See primary data	champion_FINAL_CLEAR.txt Sec. 2.7
7.2.1	9/21 A2 tasks: 0% grid despite 400 examples	verify_a2_failures.py	atomic_lora_training_summary.json, tasks_by_category.json
7.2.1	Task 694f12f3: 99.33% cell, 17.75% grid plateau	generate_smoking_gun_figure.py	atomic_lora_training_summary.json
7.3.1	Grid accuracy: 2.34% → 0.34% (re-arc to AGI-2)	arc_agi_2_analysis.py	champion_FINAL_CLEAR.txt, arc_agi_2_cross_run_summary.csv
7.3.2	Cell accuracy: 71.6% → 89.37% on AGI-2	generate_arc_agi_2_comparison.py	Same as above
7.3.3	68.6% AGI-2 failures in Low/Very Low affinity	analyze_arc_agi_2_failures.py	arc_agi_2_cross_run_summary.csv
7.4.1	S3-A: 5.68% grid; S3-B: 0.10% (57× diff.)	Manual analysis	champion_FINAL_CLEAR.txt Sec. 2.5
7.4.2	11/18 S3-A tasks show Comp. Failure pattern	generate_s3_performance_profiles.py	champion_FINAL_CLEAR.txt

B.3. Configuration and Taxonomy Files

`atomic_lora_training.yaml` Complete hyperparameter specification for all LoRA fine-tuning experiments (rank 16, alpha 32, 200 epochs with early stopping).

`split_manifest.json` Defines the 308 training / 92 validation split used throughout this work.

`data/taxonomy_classification/tasks_by_category.json` Complete mapping of all 400 re-arc tasks to their taxonomy categories.

`data/s3_subclassification/s3_final_classification.json` Sub-classification of 108 S3 tasks into S3-A (pattern-based, n=77) and S3-B (graph reasoning, n=31).

`/paper/arc_task_taxonomy.md` Canonical definitions and theoretical basis for all 9 taxonomy categories.

B.4. Figure Generation Scripts

All figures in this paper are programmatically generated from source data. The following scripts produce the main figures:

Figure 1 `figures/fig1_category_distribution.py` — Category distribution with affinity color-coding

Figure 5 `paper/generate_compositional_gap_sensitivity.py` — Sensitivity heatmap for compositional gap thresholds

Figure 6 `scripts/generate_arc_agi_2_comparison.py` — Two-panel comparison of re-arc vs ARC-AGI-2 performance

Figure 7 `scripts/generate_failure_concentration.py` — Failure concentration by affinity level on ARC-AGI-2

Figure 8 `scripts/generate_s3_performance_profiles.py` — S3-A performance scatter plot

B.5. Additional Documentation

`results/s3_subclassification/s3_generator_code_analysis.md` Rationale and code excerpts justifying the S3-A/S3-B sub-classification based on generator function analysis.

`results/generalization/arc_agi_2_evaluation.md` Complete documentation of ARC-AGI-2 evaluation protocol and results across 5 seeds.

`/taxonomy/ambiguous_tasks_analysis.md` Analysis of 14 ambiguous tasks that could not be automatically classified by the rule-based system.

Repository Structure: All file paths referenced in this appendix are relative to the root of our public reproduction package. Complete setup instructions and environment specifications are provided in the repository README.