ARC Is a Vision Problem!

Keya Hu Ali Cy Linlu Qiu Xiaoman Delores Ding
Runqian Wang Yeyin Eva Zhu Jacob Andreas Kaiming He
MIT

Abstract

*The Abstraction and Reasoning Corpus (ARC) is designed to promote research on abstract reasoning, a fundamental aspect of human intelligence. Common approaches to ARC treat it as a language-oriented problem, addressed by large language models (LLMs) or recurrent reasoning models. However, although the puzzle-like tasks in ARC are inherently visual, existing research has rarely approached the problem from a vision-centric perspective. In this work, we formulate ARC within a vision paradigm, framing it as an image-to-image translation problem. To incorporate visual priors, we represent the inputs on a "canvas" that can be processed like natural images. It is then natural for us to apply standard vision architectures, such as a vanilla Vision Transformer (ViT), to perform image-to-image mapping. Our model is trained from scratch solely on ARC data and generalizes to unseen tasks through test-time training. Our framework, termed Vision ARC (**VARC**), achieves 60.4% accuracy on the ARC-1 benchmark, substantially outperforming existing methods that are also trained from scratch. Our results are competitive with those of leading LLMs and close the gap to average human performance.*[1]

1. Introduction

Learning and abstracting concepts from a small number of demonstrations is a key feature of intelligence. The Abstraction and Reasoning Corpus (ARC) benchmark [12] was designed to incentivize machine learning research aimed at improving these capabilities. ARC consists of a collection of puzzle-like tasks (Fig. 1, top), each containing only a few examples governed by a unique underlying transformation rule. The model is expected to make predictions on each *unseen* task given a few examples. While humans are capable of solving various ARC tasks [25, 31, 32], the benchmark remains highly challenging for today's leading machine learning systems [44, 42].

The ARC problem has attracted significant attention, and substantial progress has been made in recent years [13].

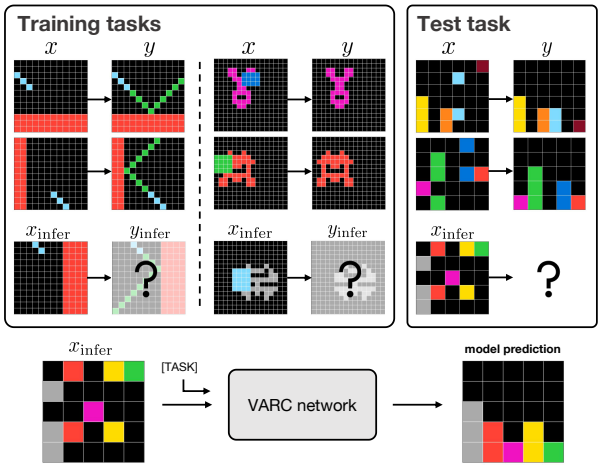

Figure 1. The **ARC benchmark** (top) consists of a collection of many different tasks, where each task has a few (*e.g.*, 2-4) examples. We propose the Vision ARC (**VARC**) framework, which addresses the ARC problem as an image-to-image translation problem, from a computer vision perspective (bottom). In this illustration, the underlying concepts of the three tasks can be roughly described by humans as: "reflection" (left), "symmetry" (middle), and "gravity" (right). These concepts are closely related to the visual and physical world.

Among a wide variety of methods, those based on large language models (LLMs) have proven highly competitive. These methods generally convert ARC inputs into sequences of text tokens for language modeling. Representative methods may involve inductive reasoning [54, 7, 50, 6], transductive reasoning [1, 19, 45], or a combination of both [35, 8, 40]. The LLMs are pre-trained on internet-scale data, from which they learn transferable common sense.

Most recently, research on *recurrent* models [53, 27] has achieved impressive results on ARC without relying on internet-scale data. These models are trained from scratch on ARC data only and perform inference through recurrent, iterative reasoning. Although they do not rely on large-scale language pre-training, these recurrent models draw strong inspiration from the success of language modeling.

Interestingly, although the ARC puzzles are typically presented visually, existing research has rarely framed ARC as a vision-centric problem. In fact, many concepts in ARC

[1] Project webpage: https://github.com/lillian039/VARC.

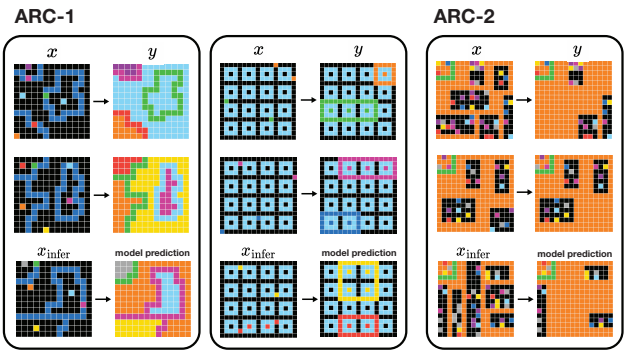

Figure 2. **Examples of unseen tasks solved by VARC.** Each panel shows an unseen test task, with demonstrations on the top and the model's prediction on the bottom. VARC correctly solves these challenging tasks.

are inherently *visual* and *physical*: *e.g.*, reflection, symmetry, and gravity, as shown in Fig. 1. Humans can solve these tasks not merely from the demonstrations, but by reasoning through analogy to their common sense obtained from external experience. Such common sense can be acquired through observing the world, particularly, the *visual* world.

Motivated by its visual nature, we approach ARC from a *vision-centric* perspective. We frame each puzzle as an image-to-image translation problem. Abstraction and inference can arise directly from visual learning, without explicit linguistic intermediates. This perspective connects ARC to classical image-to-image problems, ranging from low-level image processing (*e.g.*, [16, 43]) to high-level image understanding (*e.g.*, [38, 46]). With this connection, we can apply standard vision models (*e.g.*, Vision Transformers [17] or convolutional networks [30]) to tackle the ARC problem.

We demonstrate that incorporating *visual priors* is crucial. These priors include 2D spatial locality, translation invariance, and scale invariance. To facilitate learning these priors, we represent the inputs on a "canvas" with flexible geometric transformations, allowing the inputs to be processed as if they were natural images. A patch on the canvas can consist of exponentially many color combinations, which helps reduce overfitting and encourages the model to learn spatial priors rather than merely memorize.

With the vision-centric formulation, we train our model *from scratch* using ARC-only data. At inference time, when presented with a new, unseen task, we perform test-time training [9, 24, 49, 1, 53, 27] to adapt the model to the task, enabling it to generalize from only a few examples.

Our framework, termed Vision ARC (**VARC**), shows strong performance on the ARC benchmarks (*e.g.*, Fig. 2). VARC achieves 54.5% accuracy on the ARC-1 benchmark, using a small model with only 18 million parameters. This result substantially surpasses the best recurrent methods [53, 27] that are also trained from scratch on ARC. It is also competitive with many popular LLM-based methods. Combining VARC models through ensembling [29] further improves accuracy to 60.4%, matching the reported average human performance [31] on the ARC-1 dataset.

We hope our research will shed light on the ARC problem, and more broadly, on the field of abstract reasoning. On the one hand, the design of the ARC benchmark is based on human observations and induced rules abstracted from the visual and physical world. It is natural to explore vision-driven approaches for ARC. On the other hand, human reasoning is not confined to language or vision in isolation, but instead should integrate information across modalities. With our complementary vision-based perspective, we hope the scope of abstract reasoning will be further broadened. We invite the vision community to study the ARC problem and to advance research on abstract reasoning.

2. Related Work

Visual reasoning. Visual reasoning is a long-standing research problem. It involves not only perceiving scenes and objects, but also inferring and abstracting the relations and transformations among them. The advancement of machine learning methods has led to the development of a variety of challenging protocols, such as VQA [5, 56, 20], CLEVR [26], and Winoground [51].

The visual reasoning methods developed under these protocols typically consist of a visual perception module and a language-like recurrent module, *e.g.*, within the neuro-symbolic framework [4, 23, 3, 41]. These methods have evolved into modern vision-language models (VLMs, *e.g.*, [2, 33, 37]), in which images are converted into tokens and processed jointly with text.

Unlike ARC, classical visual reasoning protocols generally involve a training set and a test set, both of which can be viewed as instances of *the same* task. In contrast, ARC consists of a large collection of distinct tasks, each defined by only a few examples.

Approaches to ARC. Owing to the "*few-shot, many-task*" nature of ARC, LLMs have been regarded as a natural solution. A new task can be converted into a sequence of tokens, treated as a prompt, and processed by LLMs via in-context few-shot learning [55, 10]. We refer the reader to [13] for a comprehensive survey.

Recently, *recurrent* models [53, 27] have been proven effective for ARC, without the requirement of internet-scale pre-training. These models aim to mimic the hierarchical and multi-timescale processing of the human brain [53] for reasoning. At inference time, these methods adopt test-time training [9, 24, 49] on the few demonstration examples.

Related to our work, the ViT-ARC method [34] attempts to address the ARC problem using vision models. However, this method has only shown the ability to fit individual tasks in the training set; it is unable to generalize or solve any unseen test task. As such, this method has not been

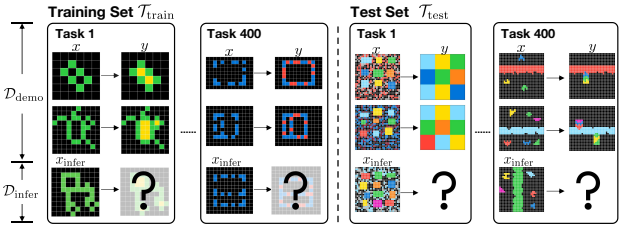

Figure 3. **The ARC problem definition**. ARC is a collection of many different tasks. For each task, a few (*e.g.*, 2-4) demonstration pairs (x, y) are given, and the model is required to infer the output from x_{infer}. The training set $\mathcal{T}_{\text{train}}$ is a collection of 400 tasks, which can be used for model training. The test set $\mathcal{T}_{\text{test}}$ contains 400 new tasks: the demo pairs of a new task are given only at inference time, based on which the model performs inference on x_{infer}.

able to satisfy the ARC protocol, whose essence lies precisely in few-shot, *cross-task* generalization. Unlike [34], our framework is designed to address the "few-shot, many-task" nature of ARC.

3. ARC as a Vision Problem

3.1. ARC Problem Definition

The ARC benchmark consists of several hundred very few-shot (*e.g.*, 2 to 4-shot) reasoning tasks. Each task, denoted by T, involves a unique underlying transformation rule, mapping from an input x to an output y. Here, x and y are both 2D grids with maximum size 30×30, in which each location has one of C different color indexes (*e.g.*, $C=10$). The ARC problem definition is illustrated in Fig. 3, which we discuss next.

A task. A "task" is the basic unit in ARC. Each task includes a few *demonstration* examples. For a demonstration pair (x, y), both x and y are known to the model. We denote the demonstration set of task T as: $\mathcal{D}^T_{\text{demo}}=\big\{(x_i, y_i)\big\}^m_{i=1}$, where m is the number of pairs (*e.g.*, m is 2 to 4). Each task T also contains a few *inference* examples, denoted as: $\mathcal{D}^T_{\text{infer}}=\big\{(x_i, y_i)\big\}^n_{i=1}$ (n is 1 or 2). At inference time, only the demo pairs $\mathcal{D}^T_{\text{demo}}$ and one input $x_{\text{infer}} \in \mathcal{D}^T_{\text{infer}}$ are given, and the model is required to infer the desired output y_{infer}.

Training set. The training set consists of multiple tasks used to train the model *offline* (*i.e.*, before a new task is given). We denote the training set as: $\mathcal{T}_{\text{train}}=\{T_i\}^k_{i=1}$, where k is the number of tasks (400 in ARC-1). Following standard machine learning protocols, samples in $\mathcal{D}^T_{\text{demo}}$ for any $T \in \mathcal{T}_{\text{train}}$ can be used for training. The "inference" samples in the training set, that is, $\mathcal{D}^T_{\text{infer}}$ for any task $T \in \mathcal{T}_{\text{train}}$, are used for validating the training process only.

Test set. The test set is a collection of *new* tasks, which are not seen during offline training. We denote the test set as: $\mathcal{T}_{\text{test}}=\{T_i\}^l_{i=1}$, with l different test tasks. Note that any test task is a "complete" and new task: that is, for any $T \in \mathcal{T}_{\text{test}}$, there also exists a demo set $\mathcal{D}^T_{\text{demo}}$, and the pairs (x, y) in $\mathcal{D}^T_{\text{demo}}$ are given to the model at inference time. The model should make use of $\mathcal{D}^T_{\text{demo}}$ to infer the output of the given x_{infer} for this new task.

The presence of new (x, y) pairs in $\mathcal{D}^T_{\text{demo}}$ at inference time allows to perform test-time training [49, 1, 9, 24], which we adopt and will discuss.

3.2. Image-to-Image Translation

With these definitions, we formulate reasoning on each task as an image-to-image translation problem. We frame the problem as per-pixel classification, analogous to the semantic segmentation problem [38].

Formally, we learn a neural network f_θ parameterized by θ. The network f_θ takes an image x_i as input, conditioned on a task token associated with the task T. The task token is represented as a learnable embedding dependent on T. The output of f_θ is a grid where each position represents a categorical distribution. The overall objective function is simply the per-pixel cross-entropy loss [38]:

$$\mathcal{L}(\theta) = \mathbb{E}_{T,i}\big[\mathcal{D}(y_i,\ f_\theta(x_i \mid T))\big]. \tag{1}$$

Here, $\mathcal{D}$ denotes the per-pixel cross-entropy loss between the ground-truth y_i and the network output.

3.3. Visual Modeling

Previous methods on ARC generally operate in the space of discrete-valued tokens, motivated by the design of language models. In our formulation of image-to-image translation, we explore *native* designs developed for vision.

Canvas. While it is straightforward to view the raw $H\times W$ grid as an $H\times W$ image, we propose more flexible transformations to represent it in a manner similar to natural images.

We define the concept of a "*canvas*". A canvas has a predefined and sufficiently large size, *e.g.*, 64×64. The raw input is transformed and placed onto this canvas. This formulation naturally accommodates translation and scale augmentations, which are common strategies for introducing translation and scale invariance in vision, discussed next. We set the background of the canvas to an additional background color, *i.e.*, the $(C+1)$-th color.

When applying a ViT model (discussed next), if we naïvely treat each raw pixel as a token, there would be only C distinct tokens. In contrast, our canvas formulation supports a much larger set of local, patch-level configurations. For example, with a patch size of 2×2 (see Fig. 5), a single patch can contain multiple colors and, in principle, has an exponentially large cardinality, $O(C^{2\times2})$. This formulation is important for improving generalization performance.

Translation and scale invariance. The "canvas" concept enables us to flexibly apply translation and scale augmentations, which are critical in standard vision models. The-

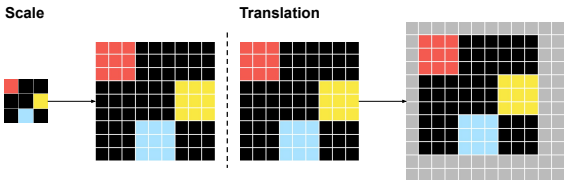

Figure 4. The raw input undergoes random scale and translation transformations and is placed on the "**canvas**" (denoted in gray).

ses data augmentations encourage the model to learn underlying mappings invariant to geometric transformations grounded in the visual world. Formally, we perform:

- *Scale augmentation*: Given a raw input, we randomly resize it by an integer scaling ratio s, duplicating each raw pixel into $s\times s$ (see Fig. 4, left). This is analogous to nearest-neighbor interpolation in natural images. However, note that "colors" in ARC do not correspond to real-world colors, so it is not meaningful to perform other interpolations (such as bilinear).
- *Translation augmentation*: given the scaled grid, we randomly place it on the fixed-size canvas. We ensure all pixels are visibile. See Fig. 4 (right).

We empirically show that these visual priors are important for generalization to unseen tasks.

Vision Transformer. Given a canvas with an input randomly placed, we perform image-to-image translation by a standard vision model. By default, we use a ViT [17].

The principle of ViT is Transformer on patches. Formally, the input canvas is divided into non-overlapping patches (*e.g.*, 2×2), projected by a linear embedding, added with positional embedding [52], and processed by a stack of Transformer blocks [52]. The model has a linear projection layer as the output, which performs per-pixel classification for each patch. Note that unlike natural images where each raw pixel has continuous values, in our case, the raw pixels have discrete values. Therefore, before patchification, we first map each pixel's discrete index into a learnable continuous-valued embedding.

Conceptually, patchification can be viewed as a special form of convolution. Like convolution, it incorporates several critical inductive biases in vision: most notably, locality (*i.e.*, grouping nearby pixels) and translation invariance (*i.e.*, weight sharing across locations).

2D positional embedding. Unlike language data, which is generally modeled as 1D sequences, images are inherently 2D. This 2D structure can be lost if we naïvely treat the embedded patches as a 1D sequence. We empirically show that explicitly modeling positions in 2D is essential.

Formally, we adopt *separable* 2D positional embeddings, following [11]: with D channels for positional embeddings, we use the first half of the channels to embed the horizontal coordinate and the second half to embed the vertical coordinate. This can be applied both to additive positional embeddings for encoding absolute positions and to the encoding of relative positions (*e.g.*, RoPE [48]).

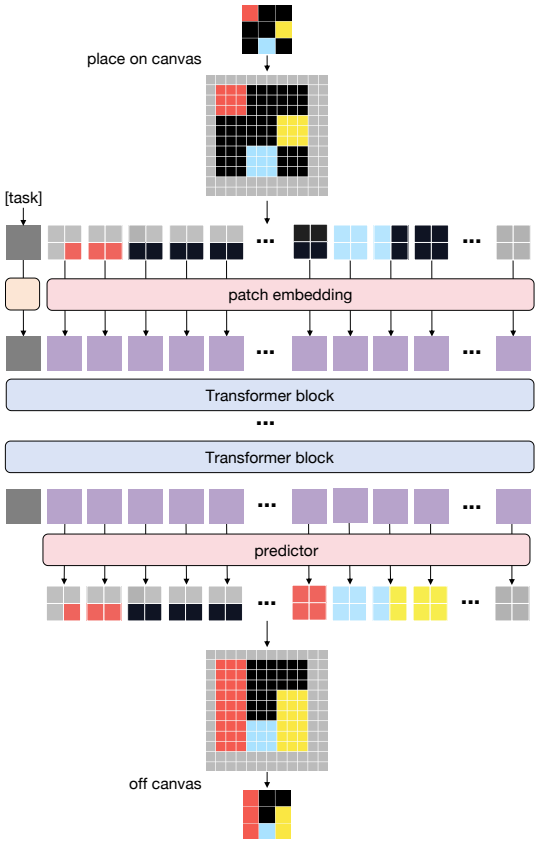

Figure 5. **The ViT architecture in VARC**. The input is randomly placed on a canvas, which is then treated as a natural image and processed by a standard ViT, conditioned on the task token.

Alternative: convolutional networks. Beyond ViT, we also study the more classical vision-based architecture, *i.e.*, convolutional neural networks [30]. Specifically, we adopt the U-Net model [46], a hierarchical convolutional network. The original U-Net was proposed precisely for the image-to-image translation problem of segmentation [46], making it a natural candidate for the problem we consider.

3.4. Two-stage Training

We adopt a two-stage training paradigm to learn the parameters of the neural network.

Offline training. This stage is applied on the entire training set $\mathcal{T}_{\text{train}}$. It is on all demos $\mathcal{D}^{T}_{\text{demo}}$ for any $T \in \mathcal{T}_{\text{train}}$. We train one model f_θ jointly for all k training tasks (*e.g.*, k=400), based on the loss in Eq. (1). All tasks share the same parameters, only except that each task has its own task-conditional token. We do not use the inference set $\mathcal{D}^{T}_{\text{infer}}$ from the training tasks (*i.e.*, $T \in \mathcal{T}_{\text{train}}$) to train the model. These sets are used only for validation purposes.

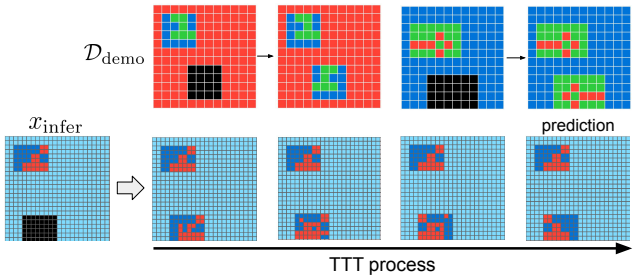

Figure 6. **Effect of test-time training.** (Top): Demonstration examples for the current task. (Bottom left): An inference example x_{infer}. (Bottom right): During test-time training, the prediction from x_{infer} becomes progressively more accurate, with the model finally generating the correct prediction.

Test-time training (TTT). Given a single new, unseen task $T \in \mathcal{T}_{\text{test}}$ from the *test* set, we perform inference by test-time training. At inference time, we are given $\mathcal{D}^T_{\text{demo}}=\{(x_i, y_i)\}_{i=1}^m$ with both input and output accessible; the model is required to make prediction for a given x_{infer} in this new task T. The test-time training followed by inference can be viewed abstractly as a function $\mathcal{F}(x_{\text{infer}} \mid \mathcal{D}^T_{\text{demo}}) \mapsto y_{\text{infer}}$.

We perform test-time training for each new task T *independently*. It has a new task token whose parameters are randomly initialized. As there are very few demo pairs in $\mathcal{D}^T_{\text{demo}}$ (*e.g.*, 2 to 4), we also perform data augmentation. We elaborate on the details in the next section and in appendix.

In summary, at inference time, the model is initialized from offline training, fine-tuned with test-time training only for the single new task T, and then performs inference on x_{infer}. As the new demo pairs in $\mathcal{D}^T_{\text{demo}}$ are very few, even with data augmentation, this test-time training process remains reasonably fast (*e.g.*, 70 seconds per task on a single GPU). Fig. 6 visualizes the effect of test-time training.

3.5. Inference

After test-time training, we apply f_θ to x_{infer} to obtain the final prediction. This process is analogous to the classical recognition problems [29, 38]. Accordingly, we adopt post-processing strategies inspired by recognition methods.

Single-view inference. Given x_{infer} and a single "view" (*i.e.*, with a given scale and translation), we place x_{infer} on the canvas and apply f_θ to predict the output. Since one output location in the raw grid may be predicted by multiple pixels on the canvas (*e.g.*, due to rescaling; see Fig. 5), we aggregate all predictions (from softmax outputs) at this location by average pooling.

Multi-view inference. It was a common practice to consolidate the predictions from multiple views (*e.g.*, see AlexNet [29]). Analogously, we adopt multi-view inference to improve accuracy, where the views are sampled with different augmentations. As the multi-view inference cost is negligible compared with test-time training cost, it is virtually nearly *free* to use many views. We use 510 random views (details are in appendix). Predictions from different views are consolidated by *majority voting* [1].[2]

Pass@2 accuracy. The ARC benchmark by default adopts the pass@2 accuracy metric: *i.e.*, two different solutions can be produced for evaluation, and a task is considered correct if one is correct. To support this metric, we adopt majority voting in multi-view inference and retain the top-2 most populated output solutions.

4. Implementation Details

We describe the major implementation choices in this section. The configuration details can be found in appendix.

Canvas. In our best-performing model, the canvas size is 64×64. In the case of ViT, the patch size is 2×2, resulting in a sequence length of 32^2. For scale augmentation, an integer scaling ratio is randomly sampled, such that the scaled grid is no larger than the canvas size. For translation augmentation, the upper-left corner is randomly sampled under the constraint that the placed image is fully visible.

Offline training. We use the standard ARC-1 training set $\mathcal{T}_{\text{train}}$ for training: it has 400 tasks with 2-4 demo pairs each. Following common practice on ARC, we also expand our training set with the RE-ARC set [22], from which we sample 1,000 additional demo pairs per task. Put together, our full training set has about 400k sample pairs. We apply translation and scale augmentation in offline training.

Test-time training. Given an unseen task $T \in \mathcal{T}_{\text{test}}$, we have 2-4 sample pairs in $\mathcal{D}^T_{\text{demo}}$. To make test-time training more feasible, we also augment the single task T into multiple *auxiliary* tasks. We do this by using standard augmentation from existing ARC methods: flip, rotation (by 90°, 180°, or 270°), and color permutation. We treat each of these test-time training augmentations as an auxiliary task, each assigned a task embedding. We also apply translation and scale augmentation in test-time training, but we do not view them as a new auxiliary task (under the assumption that all auxiliary tasks are translation and scale invariant).

5. Experimental Results

Our experiments are primarily conducted on the benchmark of ARC-1 [12]. We report the **pass@2** accuracy (referred to simply as "accuracy" hereafter) in percentage (%). To support pass@2 evaluation, we adopt multi-view inference. We also report final results on ARC-2 [14].

We evaluate our model on the ARC-1 *evaluation* set (*i.e.*, $\mathcal{T}_{\text{eval}}$). This set is conceptually a test set (see Fig. 3), but with ground truth available only for computing accuracy.

[2] In majority voting, two output grids are considered "*consistent*" only when they are identical across the entire grid. The winner is the grid that is "consistent" with the largest number of other output grids.

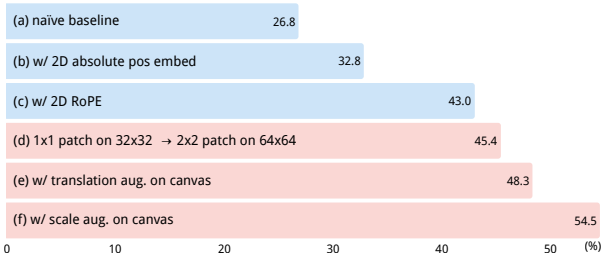

Figure 7. **Effects of visual priors in VARC**. Accuracy is reported on the ARC-1 evaluation set. The model used is ViT-18M. Entries (a-c) use a patch size of 1×1 on a 32×32 canvas, whereas entries (d-f) use a patch size of 2×2 on a 64×64 canvas. Each entry modifies the one above it. We start from a naïve baseline with components (b-f) removed. These vision priors cumulatively yield **27.7** improvement (a→f), in which the canvas-based designs (c→f) contribute an **11.5** gain.

5.1. Visual Priors

Fig. 7 summarizes the effects of visual priors, starting from a baseline (a) without the other components in this figure. These priors jointly have a gain of **27.7** points, where the canvas-based designs (c→f) has a gain of **11.5** points. We discuss these components as follows.

2D positional embedding. Extending from 1D positional embedding to its 2D counterpart is beneficial: see Fig. 7(b)(c). This is observed in both (b) absolute and (c) relative positional embeddings.

To demonstrate this effect on a stronger baseline, we replace the 2D RoPE in Fig. 7(f) with a 1D RoPE and observe a degradation of 3.5 points, from 54.5 to 51.0.

Patchification. A key design principle of our method is to prepare the input as a natural image. This enables the expansion of the token set from a very limited size (*e.g.*, 10) to an exponentially large number. The entries Fig. 7(d-f) all benefit from this design.

In Fig. 7(d), we advance from 1×1 patches on a 32×32 canvas to 2×2 patches on a 64×64 canvas. Doing so does not increase the computational cost of the Transformer. In this ablation (d), the scaling ratio is fixed as $2\times$. As such, if we constrain each 2×2 patch to cover only one raw pixel, it becomes equivalent to the 1×1 patch counterpart on the 32×32 canvas. Therefore, to ensure a meaningful comparison, we do not impose this constraint, allowing each 2×2 patch to cover *multiple* colors. This can be interpreted as one-pixel translation augmentation on the canvas.

Even so, the 2×2 patchification leads to a noticeable gain of 2.4 points, improving from 43.0 to 45.4; see Fig. 7(c,d). In spite of the small one-pixel augmentation, each patch can cover multiple colors (as in natural images), which substantially enriches the data space for learning.

Translation and scale augmentation. In image recognition, even highly capable network architectures still benefit greatly from translation and scale augmentations. We draw similar observations in ARC. See Fig. 7(e,f).

In Fig. 7(e), we apply fully flexible translation augmentation on the canvas. Compared with the "one-pixel" augmentation in Fig. 7(d), this setting yields an additional gain of 2.9 points (from 45.4 to 48.3). In Fig. 7(f), we further apply the scale augmentation enabled by the concept of canvas. Scale augmentation yields a substantial gain of **6.2** points. Unlike translation invariance, which can be partially addressed by patchification (*i.e.*, a special form of convolution), the ViT architecture has little to no inductive bias about scale invariance. This can explain why scale augmentation yields a substantial gain.

model	width	depth	#params	Gflops	acc.
ViT	384	5	6M	10	44.4
	512	10	18M	28	**54.5**
	768	20	66M	99	53.0
U-Net	setting (a)		7M	18	42.8
	setting (b)		17M	33	47.5
	setting (c)		55M	87	48.3

Table 1. **Vision backbones**. We compare variants of ViTs and U-Nets of similar sizes. U-Net settings are in appendix.

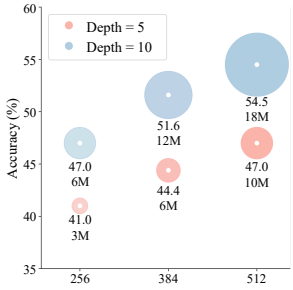

Figure 8. **Scalability**: ViTs with different width (x-axis) and depth. The circle areas denote model sizes.

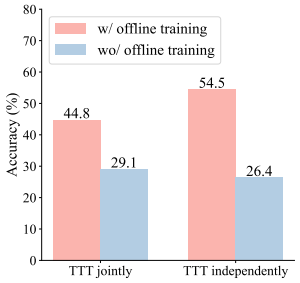

Figure 9. **TTT strategies**: with *vs*. without offline training, and joint *vs*. independent for each task.

5.2. Other Ablation Experiments

ViT *vs*. U-Net. In Tab. 1, we compare ViT with U-Nets, a type of convolutional network. We evaluate three model sizes for each architecture. Although ViTs consistently perform better, all U-Net variants achieve decent accuracy, suggesting that this problem can also be effectively addressed by classical vision backbones.

Scalability. In Fig. 8, we show ViTs with varying depths and widths. In this regime, our method demonstrates good scalability: increasing depth and/or width leads to higher accuracy as a result of better fitting. Going beyond this regime can lead to overfitting in our current setting, as

single-view, pass@1	multi-view, pass@1	multi-view, pass@2
35.9	49.8	54.5

Table 2. **Single-view *vs*. multi-view inference**.

shown in Tab. 1 for the 66M ViT model. We observe that this larger model achieves higher training accuracy, suggesting that future research should focus on generalization.

Test-time training (TTT) strategies. In Fig. 9(b), we study TTT with and without offline training, and TTT performed *jointly* on all test tasks *vs*. *independently* for each test task.

As expected, offline training greatly improves the performance of TTT, suggesting that common sense about the visual world can be learned from the training set. We also note that even without offline training, our TTT strategy can achieve nontrivial accuracy (26.4), suggesting that some tasks in this benchmark can be solved *tabula rasa*. This result outperforms that in [36] under a similar setting.

Surprisingly, performing TTT *independently* for each test task yields substantially better performance (by ∼10 points) than doing so *jointly* across all test tasks, even though the latter relies on a stronger assumption about the availability of multiple test tasks at once.[3] We hypothesize that overtraining on the test tasks may cause the model to forget the knowledge acquired during offline training.

Single-view *vs*. multi-view inference. As discussed in Sec. 3.5, we adopt multi-view inference by default. For completeness, we also examine the single-view inference accuracy. Since single-view inference cannot produce multiple predictions, we compare pass@1 accuracy. See Tab. 2.

Single-view inference has a decent pass@1 accuracy of 35.9; multi-view inference further boosts to 49.8, thanks to majority voting. Unlike typical computer vision applications such as semantic segmentation, in ARC, a mistake on even a single pixel renders the entire prediction incorrect. This may explain the large gain seen here.

5.3. System-level Comparisons

In Tab. 3 we compare with leading results using LLMs or recurrent models, on ARC-1 and ARC-2.[4]

Our model compares favorably with some of the most powerful LLMs at the time their results were reported: including Deepseek, Claude, o3, and GPT-5 (we note that given the rapid progress of LLMs, these models may have stronger results by the time our paper is public). LLMs are pre-trained on internet-scale data, and some may also incorporate multimodal data that include images. Our method does not rely on such data and uses a model that is several orders of magnitude smaller.

[3]In general, it cannot be assumed that multiple unseen tasks will be presented all at once.

system	#params	**ARC-1**	**ARC-2**
large language models (LLMs)			
Deepseek R1 [21]	671B	15.8	1.3
Claude 3.7 8k [18]	N/A	21.2	0.9
o3-mini-high [18]	N/A	34.5	3.0
GPT-5 [18]	N/A	44.0	1.9
Grok-4-thinking [18]	1.7T	66.7	16.0
Bespoke (Grok-4) [8]	1.7T	**79.6**	**29.4**
recurrent models			
HRM [53]	27M	40.3	5.0
TRM [27]	7M	44.6	7.8
vision models			
VARC	18M	54.5	8.3
VARC (ensemble)	73M	**60.4**	**11.1**
human results			
avg. human [31]	-	60.2	-
best human [18]	-	98.0	100.0

Table 3. **System-level comparisons** on the ARC-1 and ARC-2 benchmarks. LLM-based results are from the ARC-AGI leaderboard [18]. HRM, TRM, and our VARC are trained from scratch only on ARC data. Our single-model result is based on ViT, with mean±std of 54.5±0.7 (ARC-1) and 8.3±0.4 (ARC-2) over four runs. Our ensemble result aggregates an 18M ViT and a 55M U-Net, each with test-time training performed four times.

In the *controlled* setting of training from scratch on ARC data, our method substantially outperforms the recurrent models: HRM [53] and TRM [27]. Our VARC with 18M parameters is ∼10 points better than TRM on ARC-1, a >20% relative improvement. Note that, once test-time training is completed, our model performs fully *feedforward* inference, with *no recurrence* involved in reasoning.

Following the classical ensembling practice in vision (*e.g*., AlexNet [29]), we ensemble one ViT and one U-Net, each with test-time training run four times. Doing so boosts our result to **60.4**. This result closes the gap with the reported average human performance (60.2 [31]).

6. Visualization and Analysis

Beyond numerical metrics, we provide additional qualitative results that help reveal the model's behavior. We refer readers to the appendix for more visualizations.

Attention patterns. Fig. 10 shows the attention patterns of our ViT model in a test task. These attention maps show that our model can correctly reason about the relationship between a source pixel and its target pixel to copy from.

Figure 11 visualizes the layer-wise attention maps for another test task. A layer-wise map is the softmax attention map averaged across all pixels in the layer: it reveals which pixels receive the most attention in that layer. In this task, different layers exhibit different specialties: some layers at-

[4]Our ARC-2 models are trained only on the ARC-1 dataset, with test-time training and inference on the ARC-2 set.

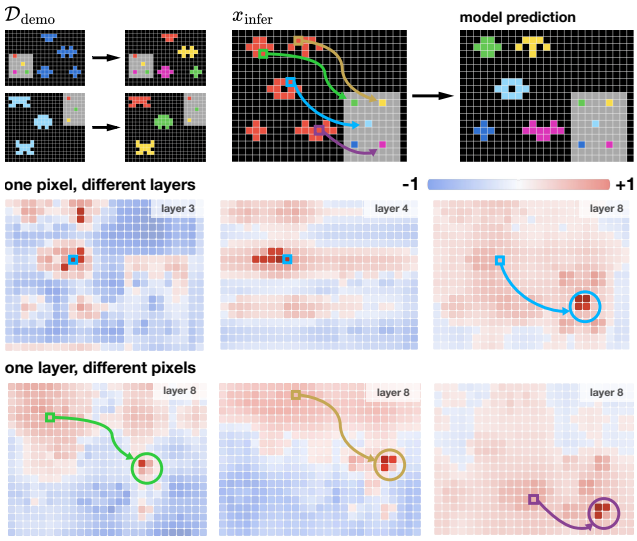

Figure 10. **Visualization of pixel-to-pixel attention. (Top)**: a test task from ARC-1 eval: showing demo pairs, inference input, and model prediction. **(Middle)**: attention maps for a single pixel across different layers. With the highlighted pixel as query, we show pre-softmax logits. Different layers exhibit different behavior. **(Bottom)**: attention maps in layer 8 with other query pixels. All of them correctly attend to their corresponding palette pixel.

tend to the pixels that are to be copied, and some layers attend to the target lines alone the eight directions.

t-SNE of task embeddings. Our model is conditioned on a task token, with an embedding learned to represent each task. With 400 training tasks in ARC-1, our model learns 400 distinct task embeddings in offline training. We visualize these 400 embeddings in the 2D space by t-SNE [39] (see Fig. 12). *Each point corresponds to a task.*.

Interestingly, we observe that nearby points in the task embedding space exhibit similar semantics. For example, the top-left corner in Fig. 12 shows two tasks related to coloring; the bottom-left corner shows two tasks related to generalized logic operations (*i.e.*, AND/OR/XOR). This visualization suggests that our method attempts to learn the *relations* between different tasks, which is an essential ability for abstraction and reasoning.

7. Conclusion

Our work explores a previously overlooked perspective in the ARC task by framing it as an image-to-image translation problem. It naturally enables the adaptation of visual frameworks and yields strong few-shot generalization competitive with recent approaches, while remaining orders of magnitude smaller than most LLM-based models. This opens up a new possibility of treating ARC as a vision-centric problem, emphasizing abstraction and reasoning emerging directly from image pixels.

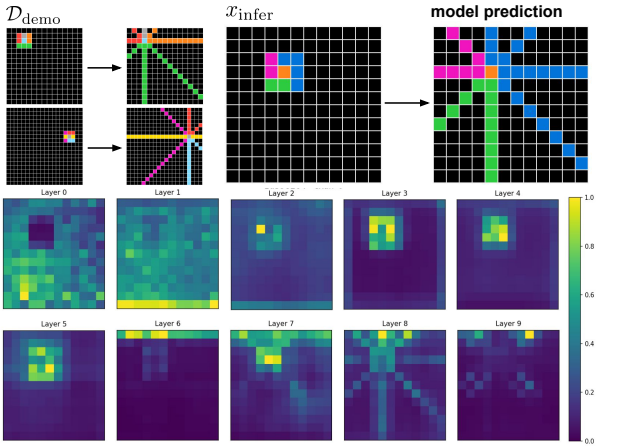

Figure 11. **Visualization of layer-wise attention maps.** For each layer, we compute pixel-to-pixel attention and then average the softmax maps across all pixels to obtain a single map per layer. This map reveals which pixels are most attended in this layer. We show a test task from ARC-1 eval. In this task, some layers exhibit strong attention to the 3×3 neighborhood, reflecting the influence of the pattern's core. In comparison, some other layers (e.g., layers 7–9) focus on the outward-radiating rays, corresponding to the rule that extends colored pixels along the eight directions.

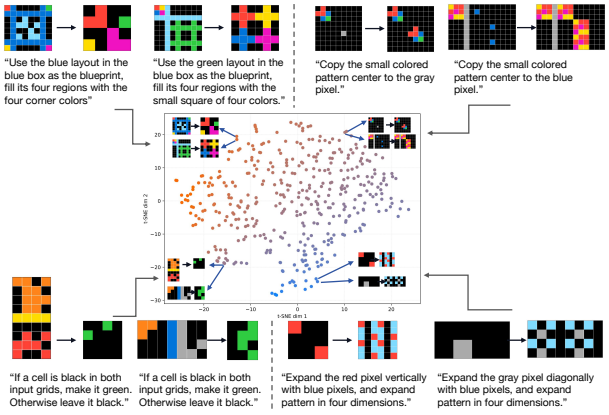

Figure 12. **t-SNE of task embeddings**, on the 400 task tokens learned from the ARC-1 training set. Each point represents a single task. To aid the reader, we provide human-written descriptions for the tasks (which are not used in any form by our method).

We hope this work will encourage the community to leverage ARC not only as a symbolic reasoning problem, but also as a testbed for promoting the generalization capacity of visual methods. Future research may extend this direction through more expressive architectures, richer visual priors, or larger-scale image pre-training. We envision that vision-centric reasoning will play a key role in building AI systems capable of learning and applying abstract concepts in a human-like manner.

References

[1] Ekin Akyürek, Mehul Damani, Adam Zweiger, Linlu Qiu, Han Guo, Jyothish Pari, Yoon Kim, and Jacob Andreas. The surprising effectiveness of test-time training for few-shot learning. In *ICML*, 2025.

[2] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob L. Menick, Sebastian Borgeaud, Andy Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karén Simonyan. Flamingo: a visual language model for few-shot learning. In *NeurIPS*, 2022.

[3] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Learning to compose neural networks for question answering. In *ACL*, 2016.

[4] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *CVPR*, 2016.

[5] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. VQA: visual question answering. In *ICCV*, 2015.

[6] Jeremy Berman. How I came in first on ARC-AGI-Pub using Sonnet 3.5 with evolutionary test-time compute. *Substack*, 2024. Accessed: 2025-10-13.

[7] Jeremy Berman. How I got a record 53.6% on ARC-AGI. *Substack*, 2024. Accessed: 2025-10-13.

[8] Jeremy Berman. How I got the highest score on ARC-AGI again swapping Python for English. *Substack*, 2025.

[9] Léon Bottou and Vladimir Vapnik. Local learning algorithms. *Neural Computation*, 1992.

[10] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *NeurIPS*, 2020.

[11] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *ICCV*, 2021.

[12] François Chollet. On the measure of intelligence. *arXiv:1911.01547*, 2019.

[13] Francois Chollet, Mike Knoop, Gregory Kamradt, and Bryan Landers. ARC Prize 2024: Technical report. *arXiv:2412.04604*, 2024.

[14] Francois Chollet, Mike Knoop, Gregory Kamradt, Bryan Landers, and Henry Pinkard. ARC-AGI-2: A new challenge for frontier AI reasoning systems. *arXiv:2505.11831*, 2025.

[15] Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion models beat GANs on image synthesis. In *NeurIPS*, 2021.

[16] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015.

[17] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

[18] ARC Prize Foundation. ARC-AGI benchmarking: Leaderboard and dataset for the ARC-AGI benchmark. https://arcprize.org/leaderboard, 2025. Accessed: 2025-11-01.

[19] Daniel Franzen, Jan Disselhoff, and David Hartmann. Product of experts with LLMs: Boosting performance on ARC is a matter of perspective. *arXiv:2505.07859*, 2025.

[20] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the V in VQA matter: Elevating the role of image understanding in visual question answering. In *CVPR*, 2017.

[21] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv:2501.12948*, 2025.

[22] Michael Hodel. Addressing the abstraction and reasoning corpus via procedural example generation. *arXiv:2404.07353*, 2024.

[23] Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. Learning to reason: End-to-end module networks for visual question answering. In *ICCV*, 2017.

[24] Thorsten Joachims. Transductive inference for text classification using support vector machines. In *ICML*, 1999.

[25] Aysja Johnson, Wai Keen Vong, Brenden M. Lake, and Todd M. Gureckis. Fast and flexible: Human program induction in abstract reasoning tasks. *arXiv:2103.05823*, 2021.

[26] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross B. Girshick. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. In *CVPR*, 2017.

[27] Alexia Jolicoeur-Martineau. Less is more: Recursive reasoning with tiny networks. *arXiv:2510.04871*, 2025.

[28] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[29] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. In *NeurIPS*, 2012.

[30] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1989.

[31] Solim LeGris, Wai Keen Vong, Brenden M. Lake, and Todd M. Gureckis. H-ARC: A robust estimate of human performance on the abstraction and reasoning corpus benchmark. *arXiv:2409.01374*, 2024.

[32] Solim LeGris, Wai Keen Vong, Brenden M. Lake, and Todd M. Gureckis. A comprehensive behavioral dataset for the abstraction and reasoning corpus. *Scientific Data*, 2025.

[33] Junnan Li, Dongxu Li, Caiming Xiong, and Steven C. H. Hoi. BLIP: bootstrapping language-image pre-training for unified vision-language understanding and generation. In *ICML*, 2022.

[34] Wenhao Li, Yudong Xu, Scott Sanner, and Elias Boutros Khalil. Tackling the abstraction and reasoning corpus with vision transformers: the importance of 2D representation, positions, and objects. *arXiv:2410.06405*, 2024.

[35] Wen-Ding Li, Keya Hu, Carter Larsen, Yuqing Wu, Simon Alford, Caleb Woo, Spencer M. Dunn, Hao Tang, Wei-Long Zheng, Yewen Pu, and Kevin Ellis. Combining induction and transduction for abstract reasoning. In *ICLR*, 2025.

[36] Isaac Liao and Albert Gu. ARC-AGI without pretraining. https://iliao2345.github.io/blog_posts/arc_agi_without_pretraining/arc_agi_without_pretraining.html, 2025.

[37] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *NeurIPS*, 2023.

[38] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[39] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of machine learning research*, 2008.

[40] Matthew V Macfarlane and Clément Bonnet. Searching latent program spaces. *arXiv:2411.08706*, 2024.

[41] Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B. Tenenbaum, and Jiajun Wu. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. In *ICLR*, 2019.

[42] Arseny Moskvichev, Victor Vikram Odouard, and Melanie Mitchell. The ConceptARC benchmark: Evaluating understanding and generalization in the ARC domain. *arXiv:2305.07141*, 2023.

[43] Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016.

[44] Rolf Pfister and Hansueli Jud. Understanding and benchmarking artificial intelligence: OpenAI's o3 is not AGI. *arXiv:2501.07458*, 2025.

[45] Jean-Francois Puget. A 2D nGPT model for ARC Prize. 2024.

[46] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015.

[47] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *NeurIPS*, 2019.

[48] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 2024.

[49] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *ICML*, 2020.

[50] Hao Tang, Keya Hu, Jin Zhou, Sicheng Zhong, Wei-Long Zheng, Xujie Si, and Kevin Ellis. Code repair with LLMs gives an exploration-exploitation tradeoff. In *NeurIPS*, 2024.

[51] Tristan Thrush, Ryan Jiang, Max Bartolo, Amanpreet Singh, Adina Williams, Douwe Kiela, and Candace Ross. Winoground: Probing vision and language models for visio-linguistic compositionality. In *CVPR*, 2022.

[52] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[53] Guan Wang, Jin Li, Yuhao Sun, Xing Chen, Changling Liu, Yue Wu, Meng Lu, Sen Song, and Yasin Abbasi Yadkori. Hierarchical reasoning model. *arXiv:2506.21734*, 2025.

[54] Ruocheng Wang, Eric Zelikman, Gabriel Poesia, Yewen Pu, Nick Haber, and Noah D. Goodman. Hypothesis search: Inductive reasoning with language models. In *ICLR*, 2024.

[55] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*, 2022.

[56] Peng Zhang, Yash Goyal, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Yin and yang: Balancing and answering binary visual questions. In *CVPR*, 2016.