

Appendix to SPHINX: A Synthetic Environment for Visual Perception and Reasoning

A. Implementation Summary

A.1. Overview

SPHINX is a framework for programmatically generating visual reasoning tasks by pairing a registry of parameterized motifs and tilings with a registry of task classes. Each task produces a complete instance consisting of (i) a rendered composite image, (ii) the exact specifications of all constituent motifs or tiles, and (iii) structured task metadata such as question, answer, reasoning type, and difficulty attributes. Task families are sampled according to configurable weights, allowing controlled variation over categories during dataset creation.

The generation engine follows a modular pipeline. A task class first samples its internal parameters (e.g., symmetry axis, sequence rule, region-of-interest) and draws motif or tiling instances from their respective registries. Motif objects expose sampling routines for geometry, appearance, and layout, while tilings produce full cell-level polygonal patches that downstream tasks query for adjacency or geometric measurements. All tasks support multiple output formats: some yield visual multiple-choice items with rendered distractor images, whereas others produce text-based options or integer-valued answers. Multiple-choice distractors are programmatically constructed to be unique and non-trivial, avoiding degenerate overlaps with the ground truth.

To further diversify the dataset, each task is associated with a bank of natural-language prompt templates. During generation, the engine selects a template, instantiates it with task-specific details, renders the composite scene at a fixed resolution, and logs all metadata: motif specifications, sampled parameters, answer, distractors, and provenance in a structured record. This design enables the construction of large-scale datasets with rich annotations, supports verifiable reward functions, and provides a unified interface for training and evaluating multimodal reasoning models.

A.2. Motif Library

SPHINX includes 25 procedurally generated visual motifs that serve as atomic building blocks to construct puzzles, scenes, and transformations. Each motif is parameter-

ized by attributes of geometry, appearance, and layout, enabling broad variation and precise control. Figure 1 shows representative instances of the 25 motifs currently implemented in SPHINX. In the following, we list the full set.

1. **Arc.** Circular arc defined by center, radius, start angle, and sweep; optionally closed into a sector. Geometry varies in radius, sweep angle, thickness, and end-cap style. Appearance includes stroke color/width, optional fills, and dashed or solid rendering. Layout covers position, rotation, and multi-arc groupings.
2. **Arrow.** Vector-like shape with a shaft and triangular or chevron head, optionally double-headed. Geometry varies via head angle, shaft width/length, curvature, and tail caps. Appearance includes filled or outlined styles, color palettes, and shading suppression. Layout controls orientation, alignment, and crowding.
3. **Bars.** Parallel rectangular bars (horizontal or vertical) used for counts or measurements. Geometry includes bar count, width/height, spacing, and jitter. Appearance supports solid or gradient fills and optional outlines. Layout covers grouping, stacking, and background grid usage.
4. **Bitgrid.** Binary on/off cell grid. Geometry includes grid size, cell aspect ratio, bit density, and mask structure. Appearance includes on/off colors, padding, rounding, and borders. Layout supports margins, rotation, and embedding within scenes.
5. **Clock.** Analog clock with ticks, numerals, and hands. Geometry varies via tick count, numeral style, and hand lengths/angles. Appearance includes face and background styles. Layout supports centering and partial occlusion.
6. **Concentric Polygon.** Multiple nested regular polygons sharing a center, optionally with rotation offsets. Geometry varies in side count, number of layers, spacing, and relative rotation. Appearance includes filled or outlined layers and alternating colors. Layout controls scale and juxtaposition.
7. **Crescent.** Lune formed by subtracting one disk from another. Geometry parameters include radii ratio, center offset, orientation, and crescent thickness. Appearance includes fill color, outline, and boundary smooth-



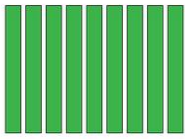
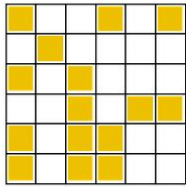



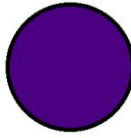
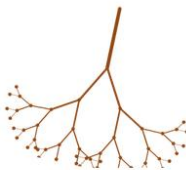








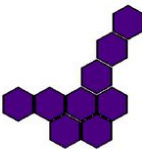


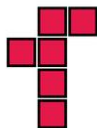




 Arc	 Arrow	 Bars	 Bitgrid	 Clock
 Concentric polygon	 Crescent	 Dot	 Fractal	 Gear
 Glyph	 Icons	 Keyhole	 Ladder	 Pictogram
 Pinwheel triangles	 Polygon	 Polyhex	 Polyiamond	 Polyline
 Polyomino	 Rings	 Segment	 Star polygon	 Stripes

Figure 1. Randomly sampled example from each Motif family.

- ing. Layout covers placement and combinations with other shapes.
- Dot.** Single filled circle or square used as a point marker. Geometry includes radius and shape type. Appearance includes color, optional halo, and stroke. Layout spans isolated dots or constellations.
 - Fractal.** Deterministic fractal curves or sets (e.g., Koch, Sierpiński). Geometry varies via generator type, iteration depth, base size, and orientation. Appearance supports stroke or filled-region modes. Layout includes centering and multi-fractal combinations.
 - Gear.** Cogwheel with teeth and a central bore. Geometry varies in tooth count, rim/hub radii, and fillet style. Appearance includes solid fills, outlines, and simplified shading. Layout supports single gears, meshing pairs, and rotations.
 - Glyph.** Typographic outlines of letters, numerals, or symbols. Geometry varies via font family, weight, character set, and size. Appearance supports filled or outlined text and anti-aliasing. Layout covers single glyphs or short strings, with rotation options.
 - Icons.** Simple pictorial symbols from a curated set. Geometry includes icon category, stroke/fill structure, and detail level. Appearance includes monochrome or

- bicolor styles. Layout supports tiling and consistent scaling.
13. **Keyhole.** Classic keyhole silhouette with circular/oval head and tapered slot. Geometry varies in head aspect ratio, slot width/length, fillets, and taper angle. Appearance includes filled or outlined styles. Layout covers orientation and grouping.
 14. **Ladder.** Two rails with evenly spaced rungs. Geometry parameters include rail spacing, rung count, rail/rung thickness, and tilt. Appearance supports stroke or filled-rectangle rendering. Layout includes vertical, horizontal, or tilted placement.
 15. **Pictogram.** Silhouette-style signs (e.g., person or restroom icons). Geometry varies by category, pose, simplification level, and aspect ratio. Appearance features filled silhouettes or minor stroke accents. Layout supports single or grid-aligned arrangements.
 16. **Pinwheel Triangle.** Radial arrangement of isosceles or right triangles forming a pinwheel. Geometry parameters include blade count, blade angle, inner radius, and chirality. Appearance includes alternating fills and optional center hubs. Layout uses centered radial symmetry.
 17. **Polygon.** Regular or irregular polygons, optionally star-convex. Geometry includes vertex count, side lengths, jitter, and rotation. Appearance supports fill, stroke, corner rounding, and hatch patterns. Layout may involve multiple polygons or packing constraints.
 18. **Polyhex.** Connected unions of unit hexagons. Geometry includes cell count, topology, and boundary complexity. Appearance supports cell outlines or merged silhouettes with fills. Layout aligns shapes to a hexagonal grid.
 19. **Polyamond.** Connected unions of unit equilateral triangles. Geometry includes cell count, triangulation orientation, and possible holes. Appearance includes outlined or merged fills. Layout aligns motifs to triangular lattice axes.
 20. **Polyline.** Open piecewise-linear path with ordered vertices. Geometry varies via vertex count, segment lengths, angle distributions, and self-avoidance. Appearance includes stroke width, joint style, and dash patterns. Layout supports start/end markers and crossings.
 21. **Polyomino.** Connected unions of unit squares. Geometry includes cell count, aspect ratio, perimeter length, and possible holes. Appearance includes borders or merged silhouettes with optional patterned fills. Layout aligns to a square grid.
 22. **Rings.** Circular annuli, single or multiple. Geometry includes outer radius, thickness, ring count, and spacing. Appearance supports filled or stroked annuli with alternating colors. Layout includes centering and multi-ring clusters.
 23. **Segment.** Straight line segment with optional end-point markers. Geometry includes length, orientation, and end-cap style. Appearance includes stroke width, color, and dash patterns. Layout spans independent or bundled segments.
 24. **Star Polygon.** Regular star polygons $\{n, k\}$ defined by step- k vertex connections. Geometry varies in n , k , radius, and rotation. Appearance includes outlined or filled modes and inner-polygon visibility. Layout supports single or layered stars.
 25. **Stripes.** Repeating parallel bands. Geometry varies in stripe width, spacing, orientation, and phase offset. Appearance includes alternating colors, gradients, and edge softness. Layout supports full-canvas or masked-region coverage.

A.3. Tilings

SPHINX provides a unified interface for generating polygonal tilings used in tile-based reasoning tasks. Each tiling generator returns a `TilingPatch` containing vertices, cells, and adjacency information, together with canonical lattice coordinates and optional Wythoffian color classes. The rendering maps each patch to a fixed-resolution canvas with consistent margins. All tilings—regular or irregular—support dual-graph construction for connectivity queries and geometry extraction via exact per-cell polygons. In addition, the tiling subsystem standardizes cell metadata, grid coordinates, and palette assignment, allowing tasks to operate uniformly across geometries. Irregular tilings use vertex pooling and quantization to maintain robust topology, while regular tilings expose symmetry-aware color classes that downstream tasks can query to form structured patterns or control distractor design.

Implementation Overview. A `TilingPatch` consists of a pooled vertex array, a list of `Cell` objects (each with vertex indices, a `kind` tag, and a grid coordinate), and optional edge records. Generators accept a `TilingSpec` that specifies `width`, `height`, and `margin_frac`. Regular tilings (square, triangular, hexagonal, rhombille) implement canonical Wythoffian colorings, while irregular tilings (e.g., circles) rely on uniform or heuristic palettes. The dual graph is built through `build_dual_graph`, where nodes are cells and edges represent shared boundaries (or, optionally, shared vertices when `connect_on_touch` is enabled).

Implemented Tilings We implement the following five tilings:

1. **Circles (Circle Packing).** Equal-radius disks are placed on a triangular lattice and approximated by m -gons whose vertices align with the six tangential directions. Grid coordinates index lattice centers. Because neighboring polygons often touch at single vertices, adjacency uses `connect_on_touch=True`.

This tiling does not expose Wythoffian color classes. Variants include grid size, radius, polygon fidelity, and margins.

2. **Square.** Axis-aligned unit squares arranged on an integer lattice form a standard `width × height` rectangular grid. Each cell is indexed by its lower-left (i, j) coordinate. Adjacency is defined by shared edges, and a four-class Wythoffian coloring is given by $((i \& 1) \ll 1) | (j \& 1)$. Variants include grid dimensions, margins, and rendering styles.
3. **Triangular (Equilateral).** Point-up and point-down equilateral triangles form a row/column lattice, with alternating rows shifted by half a cell horizontally. Cells are indexed by integer coordinates consistent with this construction. Adjacent cells share full edges, and a three-class Wythoffian coloring is provided by $(i + 2j) \bmod 3$. Variants include grid size, triangle side length normalization, and margin.
4. **Hexagonal (Flat-Top, odd- q).** Regular flat-top hexagons are arranged on an odd- q axial grid. Each hexagon is generated from a fixed radius using six vertices spaced at 60° . Cells have up to six neighbors via edge adjacency. A three-class Wythoffian coloring is obtained from $(q - r) \bmod 3$. Variants include grid dimensions (axial columns/rows), hex radius, and margins.
5. **Rhombille.** Each hexagon of a flat-top hex grid is subdivided into three congruent $60^\circ/120^\circ$ rhombi. Cells are indexed by axial (q, r) plus a sector identifier $(0, 1, 2)$. Vertex pooling ensures a shared center for the three rhombi. Adjacency connects rhombi both within each subdivided hex and across boundaries of the underlying hex grid. A Wythoffian three-class coloring is available via axial parity. Variants include grid size, radius scaling, and margins.

Notes for Downstream Tasks. Connectivity-based tasks operate on the dual graph; shortest-path tasks use BFS or 0-1 BFS since edges are unweighted. Geometric and counting tasks use `cell_polygons()` for exact polygon shapes, with vertex pooling providing numerical stability in irregular tilings.

B. Task Descriptions

B.1. Geometric Reasoning

Figure 2 shows examples of this type of task.

B.1.1. Positional Count

Problem. Positional counting relative to non-overlapping reference shapes (rectangles, circles, and triangles). The objective is to count small shapes that satisfy a strict spatial relation to a chosen reference.

Construction. Place 1-4 large reference shapes with

enough separation. Sample small shapes (circle, triangle, square, pentagon, hexagon) with pairwise non-overlap and strict visual separation from all reference boundaries. Evaluate strict radius-aware predicates (inside, outside, above, below, left, right) to form the label.

Variants. Six relation categories crossed with multiple small-shape kinds; background and counts vary with seed.

Complexity. We measure complexity with the count of larger reference shapes.

Answer type. Integer count.

B.1.2. Shape Sorting

Problem. Ordinal sorting over labeled geometric primitives under a specified metric.

Construction. Sample a family (polygon, ellipse, angle, line) and a metric (polygon/ellipse area or perimeter; angle measure; line length). Sample values with a minimum relative gap and render using a random-pack layout with uniform-height font for the label.

Variants. Four families with metrics as above; the number of items k is drawn from configurable bounds.

Complexity. We measure complexity with the number of items sampled k .

Distractors. Multiple-choice over comma-separated orderings; distractors are produced by swapping two positions in the true ordering (with random shuffles as a fallback to maintain four unique options).

B.1.3. Stack Count

Problem. Given overlapping sheets of equal area, count small objects that lie strictly inside a designated sheet (excluding the topmost one).

Construction. Choose a stack type (rectangle, circle, equilateral triangle). Generate k sheets with controlled pairwise overlap ratios and identical area; draw small objects (circle, triangle, square) on top of the stack. Pose an inside-of-border query about an occluded sheet.

Variants. Three stack families \times three kinds of small-objects. The prompts vary in target sheet (color) and object kind.

Complexity. We measure complexity with the number of large stacked shapes.

Answer type. Integer count.

B.1.4. Pie Chart

Problem. Ordinal reasoning over a single pie chart. The model must rank categories by slice size (ascending or descending) without access to numeric labels.

Construction. Sample k categories with percentages that meet a strict relative gap; optionally, derive consistent integer counts for provenance. Render a legend-only chart.

Variants. Two variants induced by the crossing of sort direction (ascending/descending, 50/50).

Complexity. The number of k categories is used to measure complexity.

Distractors. Multiple-choice over comma-separated rankings; distractors come from swapping two positions in the true ordering (with random shuffles as a fallback to keep four unique options).

B.1.5. Chart Comparison

Problem. Matching of proportions on two charts. A top chart (pie or bar) defines the color \rightarrow percent mapping; the set of options comprises four options of the opposite chart type. Exactly one option preserves the mapping.

Construction. Sample k categories, distinct integer percentages for the categories that sum up to 100, and a distinct color palette.

Variants. Two display regimes with the top chart as a pie chart or a bar chart and the options as the opposite chart type.

Complexity. We measure complexity with the number of k categories in the charts.

Distractors. Wrong options are produced by jittering and/or permuting the percentage vector. Candidates are admitted only if they pass absolute/relative difference thresholds and pairwise image-level distinctness checks.

B.2. Counting

Figure 3 shows examples of this type of task.

B.2.1. Venn Diagram

Problem. Inclusion/exclusion over axis-aligned shapes with per-region numeric labels.

Construction. Sample 2-4 axes-aligned rectangles with a connected union. Induce a partition grid, place one integer in each non-empty atomic region (with skinny-region fallbacks), and pose include/exclude queries whose truth set uniquely determines the sum.

Variants. Four query modes sampled from (“only”, “intersection”, “butnot”, “union”); include/exclude masks are resampled until the target region is unique and non-empty.

Complexity. We measure complexity with the number of rectangles.

Answer type. Integer sum.

B.2.2. Shape Counting

Problem. Counting of sub-shapes (rectangles, squares, triangles, parallelograms) within a single connected figure.

Construction. Draw one connected figure using one of several generators (axis-aligned polyomino, skewed poly-parallelogram, irregular/regular grids, staircase, triangular lattice, inscribed overlay). Render on a plain white background and compute the ground-truth count using exact combinatorial routines matched to the generator.

Variants. Eleven generator families (as above), each paired with appropriate query types. Instances are only emitted when the computed answer lies within configured bounds.

Complexity. The number of shapes in a figure (answer).

Answer type. Integer count.

B.2.3. Tiles Line Length

Problem. Edge-step length estimation for a highlighted colored polyline.

Construction. On a chosen tiling, sample k non-overlapping polylines, record their lengths, and ask for the length of one specified by color.

Variants. Four different types of tilings (square, triangular, hexagonal, rhombille).

Complexity. The number of cells in the tiling.

Answer type. Integer length.

B.2.4. Tiles Line Intersections

Problem. Intersection counting over colored polylines constrained to tile edges.

Construction. Build a vertex graph for the selected tiling; lay out k vertex-simple polylines with distinct colors and no shared edges.

Variants. Two different types of tiling (square, triangular).

Complexity. The number of intersections measures difficulty.

Answer type. Integer number of shared vertices (including endpoints).

B.2.5. Tiles Recoloring

Problem. Cell-wise recoloring/difference counting between two related boards.

Construction. Grow a connected region on the left board; derive the right board by adding/removing a connected set (same-color variant) or additionally recoloring overlap (color-change variant).

Variants. Two variants - same color vs. color change - across several tiling families.

Complexity. The number of different cells measures the difficulty.

Answer type. Integer number of differing cells.

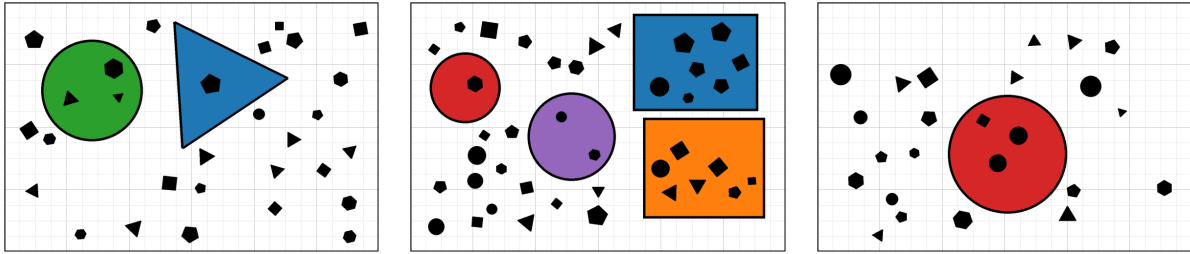
B.3. Symmetry & Pattern Recognition

Figure 4 shows examples of this type of task.

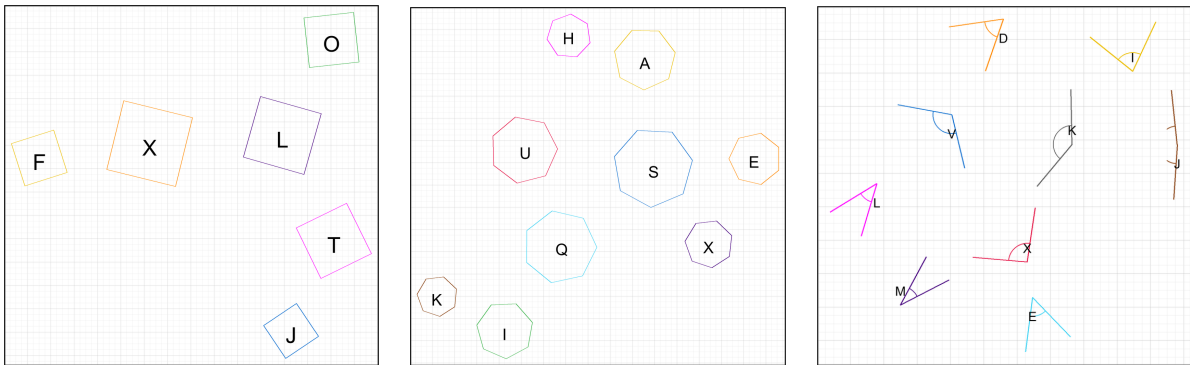
B.3.1. Mirror Identification

Problem. Textual classification of mirror symmetry (including “none”) for a composite scene.

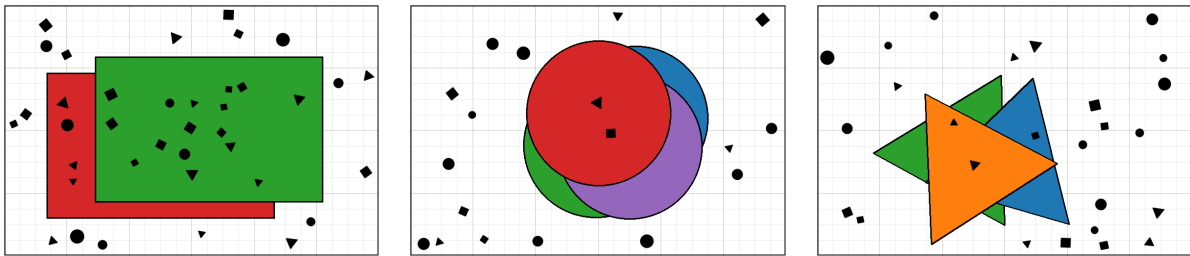
Construction. Place motif instances inside class-specific fundamental regions to synthesize scenes. Verify the final bitmap’s category via color-aware symmetry tests; pair with six textual options and shuffle.



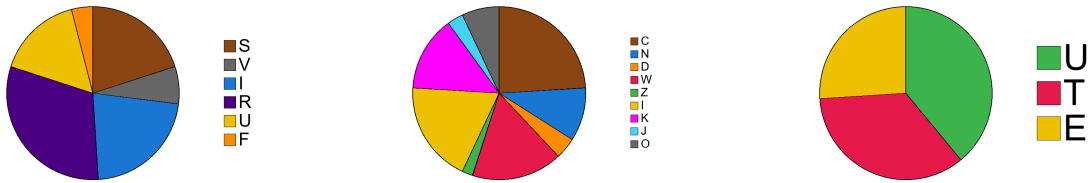
(a) **Positional Count** Count the small shapes that satisfy a specific spatial relation to a larger shape.



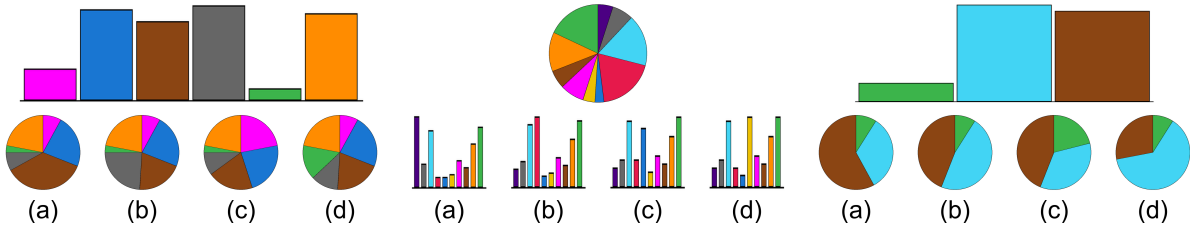
(b) **Shape Sorting** Sort the labeled shapes by a given metric, such as area or angle.



(c) **Stack Count** Count the number of a certain small shape that are fully inside one of the occluded, overlapping sheets.



(d) **Pie Chart** Rank the slices of the pie chart by size.



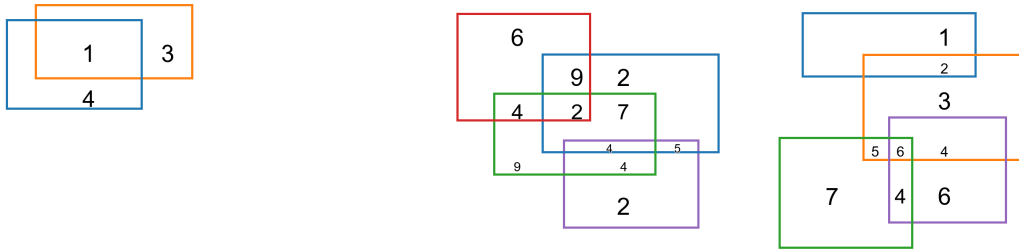
(e) **Chart Comparison** Find the bar/pie chart that correctly represents the proportions in the top chart.

Figure 2. Examples of Geometric Reasoning and Chart tasks.

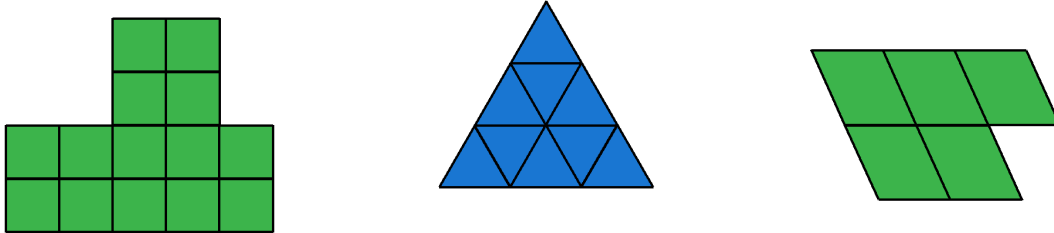
Variants. Six labels - vertical, horizontal, main diagonal, anti-diagonal, vertical+horizontal, none - with target

count and canvas scale adapted to the class.

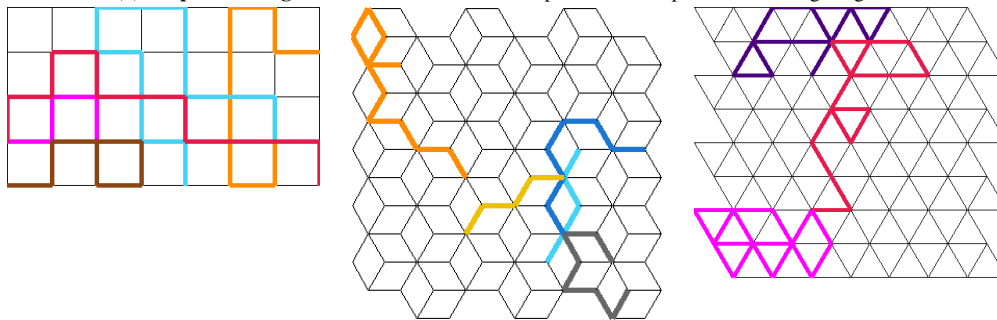
Complexity. Not measured.



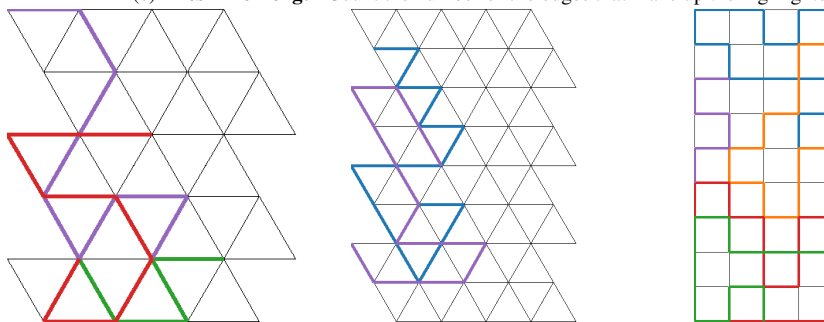
(a) **Venn Diagram** Calculate the sum of numbers in specified regions in the Venn diagram.



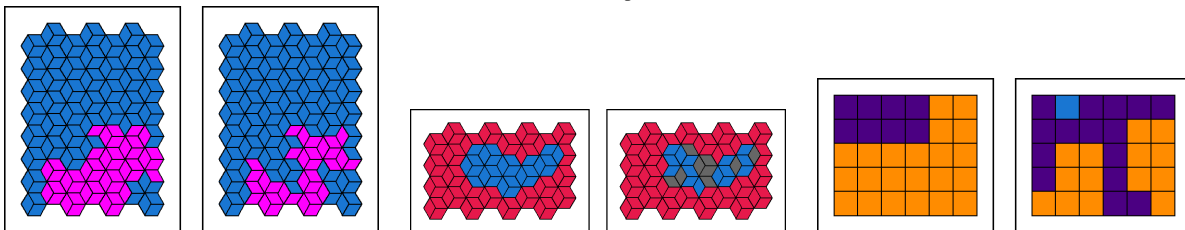
(b) **Shape Counting** Count the total number of a specific sub-shape within the larger figure.



(c) **Tiles Line Length** Count the number of tile edges that make up the highlighted line.



(d) **Tiles Line Intersections** Count the number of points where the two colored lines intersect.



(e) **Tiles Recoloring** Count the number of cells that have different colors between the two boards.

Figure 3. Examples of Counting tasks.

Distractors. The five incorrect textual descriptions serve as distractors; all six labels are offered.

B.3.2. Symmetry Fill

Problem. Grid completion under a specified mirror constraint. A 2×2 grid is shown with one missing tile; select the tile that restores the target symmetry.

Construction. Render a base tile, apply the rule (vertical, horizontal, both, main-diagonal, anti-diagonal) to fill the grid, remove one tile, and construct options by applying distinct transforms while enforcing pairwise distinctness.

Variants. Five rule keys as above; the missing position and motif vary.

Complexity. Not measured.

Distractors. Transform pool filtered to retain only visually distinct candidates; select three and shuffle with the correct transform.

B.3.3. Frieze Groups

Problem. Odd-one-out identification among four horizontal strips, each generated from a frieze symmetry; three share the same neighbor rule, one differs.

Construction. Sample a motif family; choose a majority frieze group for three strips and a distinct group for the odd strip. Render with consistent spacing and label (a-d).

Variants. Six Conway frieze groups (step, sidle, jump, spinning hop, spinning sidle, spinning jump). The strip length and option order vary per instance.

Complexity. Not measured.

Distractors. The distractors are simply additional strips from the majority frieze class; the odd class is unique by construction.

B.3.4. Wallpaper Groups

Problem. Odd-one-out among four 2D wallpaper patches; three are sampled from one wallpaper group and one from another.

Construction. Sample a motif family and wallpaper groups; generate patches under each group, crop to equal square tiles, and compose a labeled 2×2 grid.

Variants. Seventeen IUC wallpaper groups; the majority/odd selection and the option order are randomized.

Complexity. Not measured.

Distractors. The three majority-group patches form the distractor set by construction.

B.4. Sequence & Transformation Reasoning

Figure 5 shows examples of this type of task.

B.4.1. Transform Result Identify

Problem. Visual selection of the result of applying a sampled transform to the original tile.

Construction. Render a motif patch, center it on graph paper, sample a transformation, and construct one correct

and three incorrect image options with consistent placement and borders. Compose a top/bottom layout with labels.

Variants. Eight transformation families; translations use randomized vectors.

Complexity. Not measured.

Distractors. Render alternative transforms (including alternative translation vectors) and retain only candidates that are pairwise distinct.

B.4.2. Transform Pair Infer

Problem. Identify the single transformation that maps a source tile to a target tile; “none of the above” may be correct by omission.

Construction. Render a motif on graph paper, choose a true transform from mirrors/rotations/translation, synthesize the target, and verify uniqueness against the full rule set. Compose a side-by-side display with an arrow and six labeled textual options.

Variants. Up to eight answer classes: seven concrete transforms (vertical mirror, horizontal mirror, main diagonal mirror, anti-diagonal mirror, 90° rotation, 180° rotation, 270° rotation, translation) plus none (correct with probability $1/6$ when the true transform is withheld).

Complexity. Not measured.

Distractors. When the true transform is present, sample other transforms as distractors with uniqueness filtering; when omitted, append none and select the remainder accordingly (with none fixed to the final slot for clarity).

B.4.3. Transform Similarity Identify

Problem. Similarity-based selection under Euclidean similarity (uniform scale + D_4 rigid/mirror motions). Either select the single similar option or the single dissimilar one.

Construction. Render an asymmetrical motif and produce options using the allowed D_4 transformations with optional uniform scaling and translation. For “dissimilar”, apply enabled breaker warps (e.g., anisotropic scale, shear, perspective) and reject near-similar outcomes via a canonical checker.

Variants. Two core variants with four options.

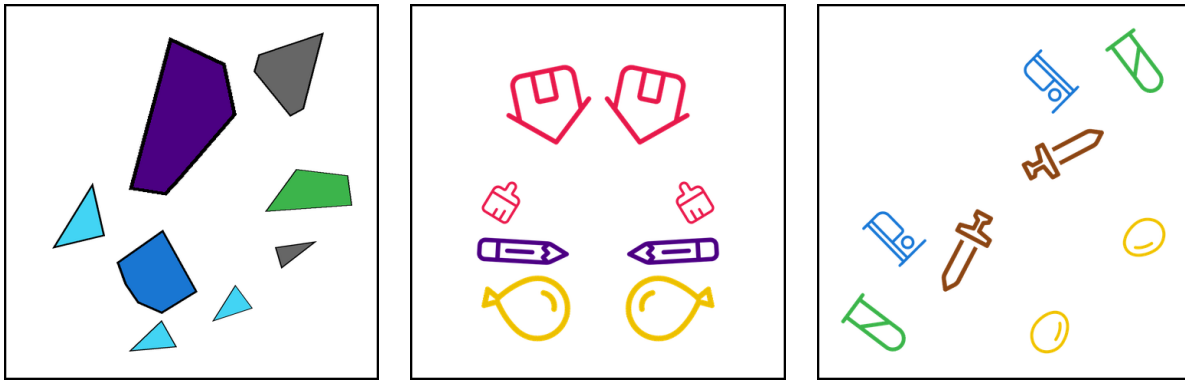
Complexity. Not measured.

Distractors. For “similar”, distractors are other outcomes that remain distinct; for “dissimilar”, distractors are similar options.

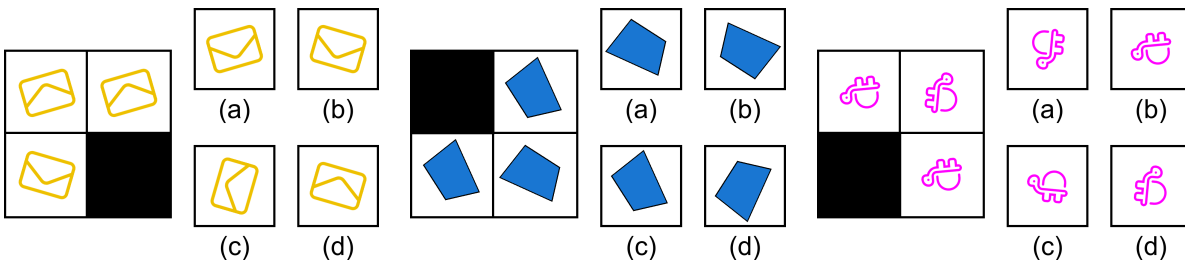
B.4.4. Sequence Rotation

Problem. Rotation-only progression over a single bitmap with a constant angular step; one panel is masked.

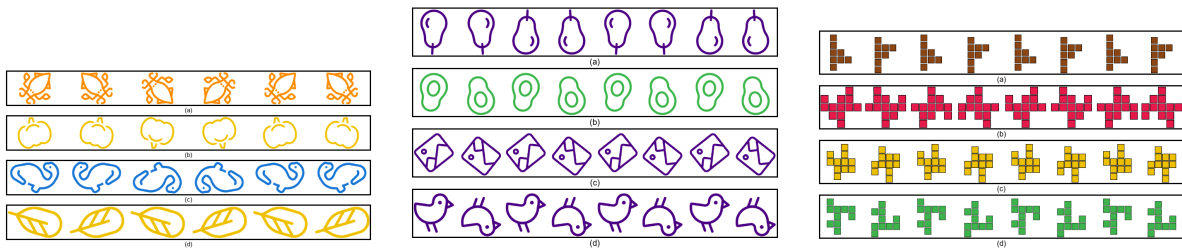
Construction. Render a base motif, compute a global scale fitting all sampled rotations, and generate tiles using a step from $\{30^\circ, 45^\circ, 60^\circ, 90^\circ\}$ in either direction. Mask one panel and present four options.



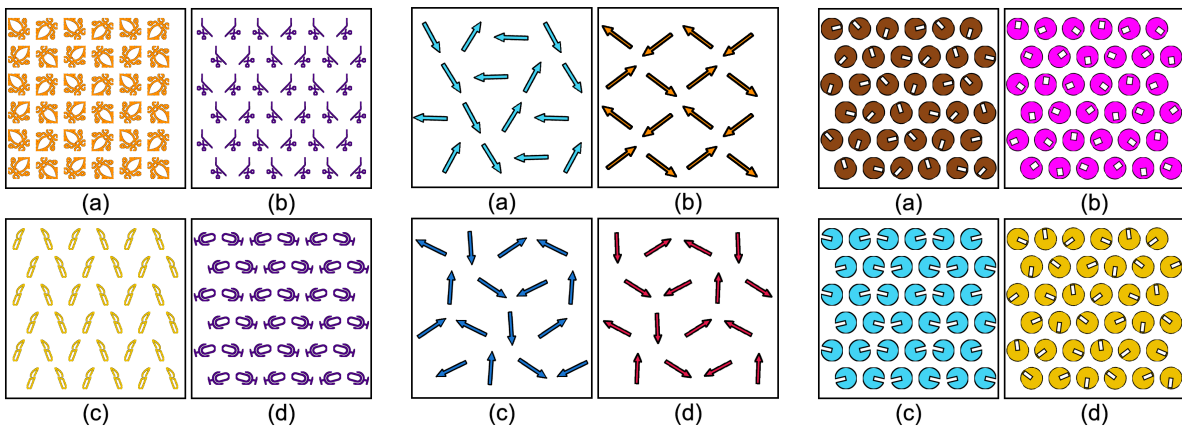
(a) **Mirror Identification** Identify the axis of mirror symmetry in the image if there is one.



(b) **Symmetry Fill** Choose the tile that completes the grid according to the specified symmetry rule.

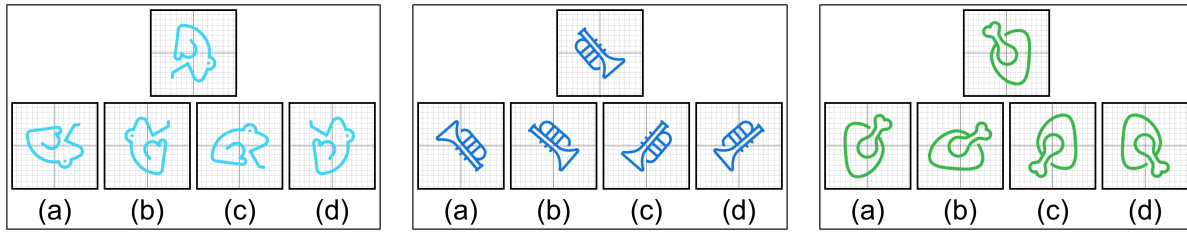


(c) **Frieze Groups** Identify which of the four patterns belongs to a different frieze symmetry group.

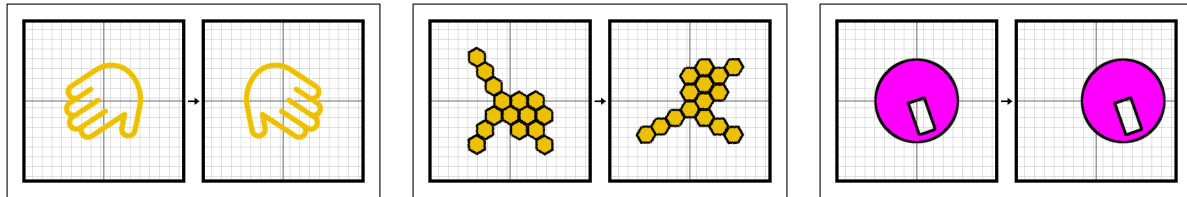


(d) **Wallpaper Groups** Identify which of the four patterns belongs to a different wallpaper symmetry group.

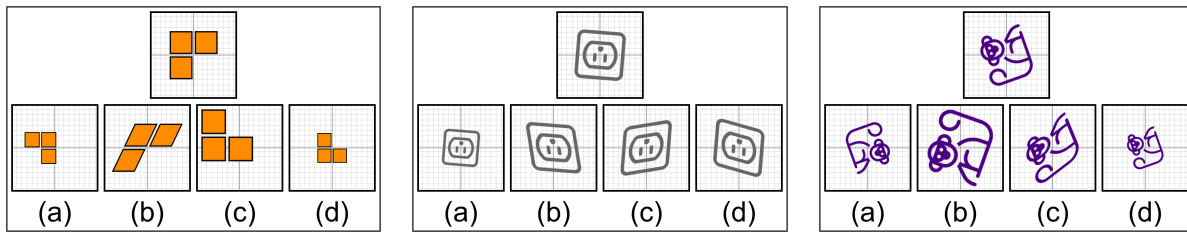
Figure 4. Examples of Symmetry tasks.



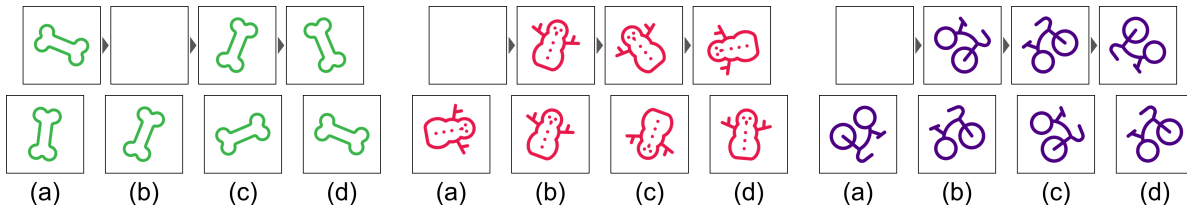
(a) **Transform Result Identify** Choose the image that shows the correct result of applying the given transformation.



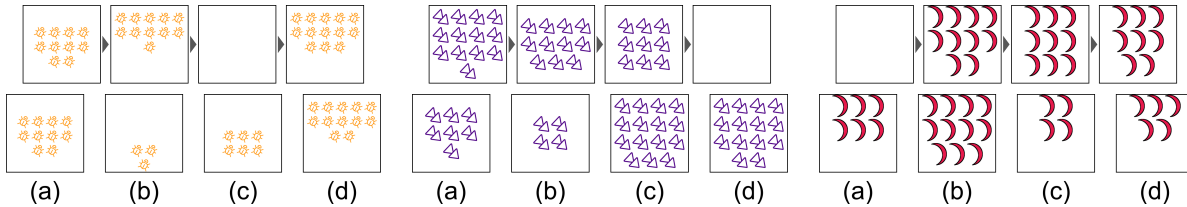
(b) **Transform Pair Infer** Identify the transformation that maps the left image to the right image.



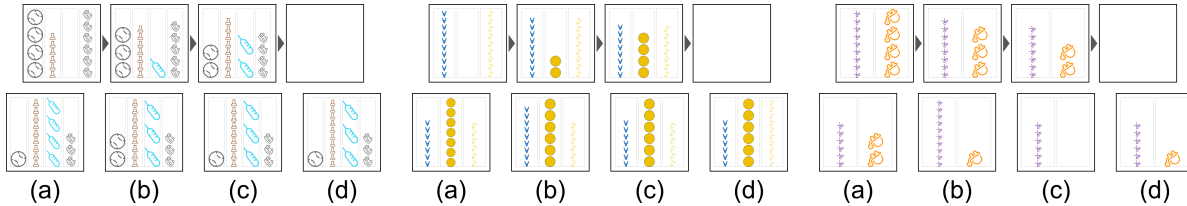
(c) **Transform Similarity Identify** Find the shape that is a similar version (rotated, scaled) of the target.



(d) **Sequence Rotation** Complete the sequence by finding the missing panel, which follows a constant rotation.



(e) **Sequence Arithmetic** Complete the sequence, which follows an arithmetic progression of shapes.



(f) **Sequence Multi-Column Arithmetic** Predict the final panel where each column follows its own progression.

Figure 5. Examples of Transformation and Sequence tasks.

Variants. Eight rotation regimes (four step sizes \times two directions); mask index is uniform.

Complexity. Converts the rotation step into a reversed ordinal (i.e., a smaller rotation difference is more com-

plex).

Distractors. Alternative rotation angles filtered by separation thresholds; weakly separated candidates are rejected.

B.4.5. Sequence Arithmetic

Problem. Next-step prediction in a count-based progression with one masked panel.

Construction. Sample a motif by weights. Draw a sequence with the count changing by a set increment/decrement; mask one panel and provide four choices.

Variants. Four candidate step choices (0, 1, 2, 3), either increasing or decreasing.

Complexity. Maximum number of shapes across the sequence.

Distractors. Different incorrect counts are made and checked for enough visual difference from other options.

B.4.6. Sequence Multi-Column Arithmetic

Problem. Multi-column next-step prediction where each column follows its own arithmetic progression.

Construction. Sample 2-6 columns, motif kinds, and per-column base specs; draw four time steps using a shared within-column scale set by the maximum count. Hide the final panel and provide four candidates for continuation.

Variants. Continuous parameterization of column count, motifs, and steps.

Complexity. The number of columns is used to measure complexity.

Distractors. Edit exactly one column per wrong option, escalating $\pm\Delta$ until the local change exceeds a threshold; reject duplicate/low-contrast candidates.

B.5. Topological & Graph Reasoning

Figure 6 shows examples of this type of task.

B.5.1. Tiles Geometry

Problem. Geometric measurement over colored regions on a tiling (area, perimeter, holes, area difference, union perimeter).

Construction. Sample a tiling, paint disjoint regions, compute region graphs, and evaluate the requested measure. Render a white board with a natural-language prompt.

Variants. Five query types: single region area, single region perimeter, single region hole, two region area difference, union of two region perimeter, with per-instance color selection.

Complexity. The size of the tiling is the measure of complexity.

Answer type. Integer.

B.5.2. Tiles Connected Component

Problem. Component analysis on a colored tiling. Query the size of the largest/smallest component or the number

of components within a specified color under a given adjacency notion.

Construction. Sample a tiling and a non-uniform coloring; build the dual graph with edge adjacency (or point-touch for circular tilings). Compute per-color connected components and select a query with a unique answer (enforced for extreme queries).

Variants. Three different types of measures (largest size, smallest size, count components) on five different tilings.

Complexity. The number of components measures complexity.

Answer type. Integer.

B.5.3. Tiles Shortest Path

Problem. Shortest-path computation on a cell graph with obstacles; return the minimum number of edge-steps or -1 if unreachable.

Construction. Sample a tiling, build the dual graph, sample an obstacle field from beta-regime priors (sparse, dense, balanced, patchy), choose start/end tiles, and use BFS to verify distance or enforce unreachable cases.

Variants. Reachable (0.9 probability) vs. unreachable (0.1 probability) variety, five different tilings.

Complexity. The size of the tiling.

Answer type. Integer (distance) or -1 if unreachable.

B.5.4. Missing Tiles

Problem. Completion of a partially blanked tiling via color restoration or shape fitting (orientation changes allowed).

Construction. Sample a tiling and remove a connected region of bounded size. In the color variant, recover the exact color assignment for the missing cells. In the shape variant, recover the exact shape up to the tiling's dihedral symmetries.

Variants. Two balanced variants (color vs. shape) across four tilings (square, triangular, hexagonal, rhombille).

Complexity. The size of the tiling is used to measure complexity.

Distractors. Color variant performs pairwise color swaps or Dirichlet-weighted palette shuffles; shape variant samples alternative connected subsets of equal size that are non-congruent under allowed symmetries.

B.5.5. Tiles Composition

Problem. Piece equivalence and assembly. Either decompose a connected region into a multiset of connected pieces (bags), or compose a bag into a single connected target.

Construction. Sample a tiling and connected region; split into 2-4 connected pieces via randomized BFS growth. In “decompose”, show the region on top and candidate bags below; in “compose”, show a bag on top and candidate target shapes below. Normalize framing across options.

Variants. Two modes (decompose vs. compose) \times two color modes (uniform, random_per_cell). Additional variation from piece counts and tiling families.

Complexity. The number of connected pieces is used as a measure for complexity.

Distractors. For decompose, bags reuse piece cardinalities but alter piece shapes. For compose, candidates match area but do not correspond to the true union of pieces.

C. Human Evaluation

We conducted a human evaluation using a custom-built web application. The participants accessed the app through a browser and were assigned a set of 25 problems (or 10 problems if they explicitly chose the shorter version). Each problem consisted of a visual prompt (image and/or text) and an input field for responses. In total, we collected 32 completed evaluations.

The application enforced basic validation (e.g., number formats, single-choice letters, or ordered lists) to ensure that responses were well-formed. For each participant, we record the following.

- Response text
- Correctness (with respect to the ground truth)
- Per-question time taken
- Overall completion time
- Types of tasks assigned

To reduce variability in prior knowledge, the interface also provided a dedicated *Definitions* panel containing concise explanations of key terms and concepts (e.g., symmetry, rotation, translation). This feature ensured that all participants could engage in tasks from a comparable baseline of conceptual understanding, thereby minimizing confounding effects due to varying background knowledge.

After completing the problem set, participants completed a *post-questionnaire survey* in which they rated perceived difficulty, clarity, familiarity, and participation, along with providing optional feedback.

Human Evaluation Setup Figure 7 shows the web interface used to collect human responses to the specific tasks assigned for SPHINX.

Human Performance Analysis Figure 8 shows human performance in evaluation tasks, highlighting time–accuracy analysis, and the relationship between difficulty ratings and accuracy.

D. Analysis of GPT-5 Response

D.1. Shape Counting

Shape counting is the task on which GPT-5 most strongly surpasses human performance (76% vs 55.2%). Figure 9 presents two correct model responses in the top row and two incorrect responses in the bottom row. In the correct cases, GPT-5 produces clear, step-by-step reasoning: it enumerates all relevant sub-shapes, organizes them by size or configuration, and adds the totals in a structured way. The textual reasoning is coherent and reflects an understanding of how larger shapes are made up from smaller ones.

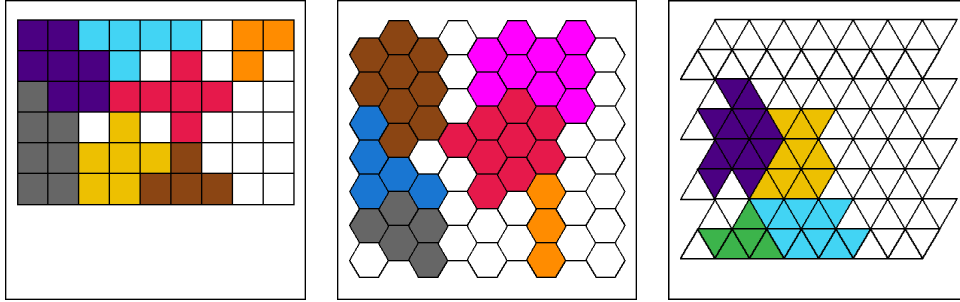
The incorrect examples reveal a different pattern. GPT-5 often reasons correctly in text, but the reasoning is anchored to an inaccurate visual extraction. In example (iii), the model incorrectly states that the figure contains five rows when it actually contains four, and all subsequent steps follow from this mistaken premise. In example (iv), the model identifies five parallel lines from left to right instead of six and then applies a suitable combinatorial formula based on that incorrect count. The final numeric answer would have been correct if the model had extracted the correct number of lines.

These observations also help explain why the human accuracy is lower than that of GPT-5. The participants viewed the images on a monitor and performed all the counting mentally without any external notes. Since shape counting requires keeping track of several intermediate quantities, arithmetic slips or overlooked configurations are common even when the visual perception itself is correct.

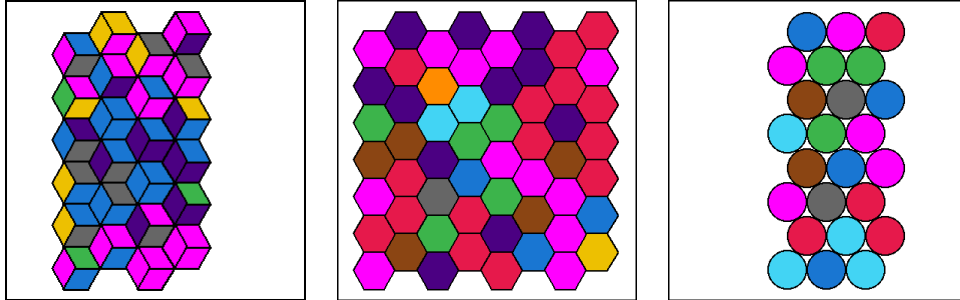
D.2. Tiles Composition

Tiles Composition is the task with the second-highest human accuracy, and GPT-5 exceeds human performance by a small margin (about 2%). The task itself is non-trivial, even for humans. GPT-5 often produced answers without providing explicit reasoning. Of 100 questions, 30 responses from the API did not contain visible reasoning. Figure 10 shows representative examples, with two correct responses in the upper row and two incorrect responses in the lower row.

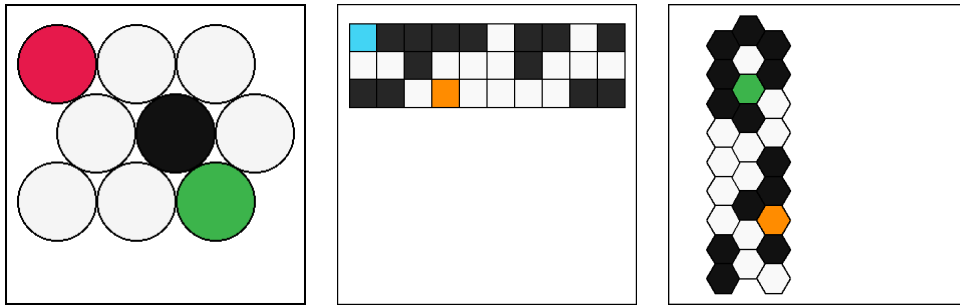
In the correct cases, GPT-5 sometimes reached the right answer despite imperfect or partially incorrect reasoning. For instance, in example (i), the model attempted to compare the option tiles by counting the total number of small squares. Although its counts were inconsistent across the choices, it nevertheless provided a correct explanation afterward, describing how the smaller pieces can be assembled to form the larger target shape. This suggests that the model is capable of meaningful geometric reasoning even when the initial steps of its analysis are flawed. Example (ii) further illustrates this. There,



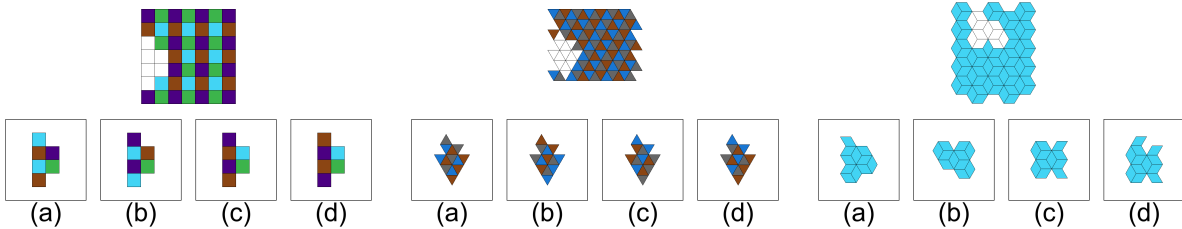
(a) **Tiles Geometry** Calculate a geometric property, such as area or perimeter, of a specified region.



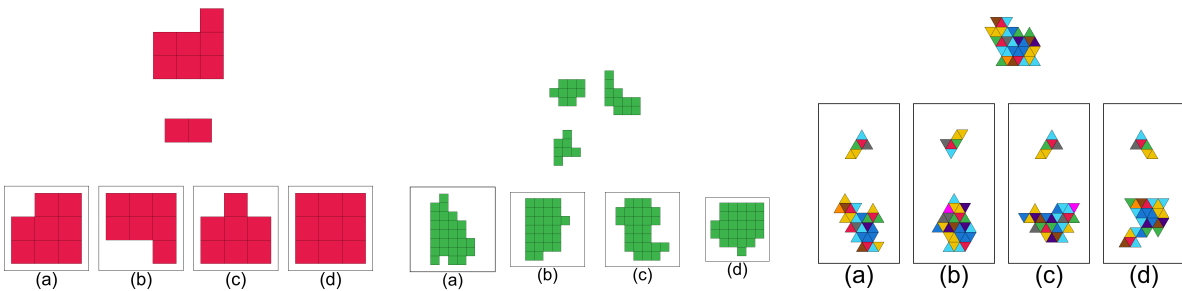
(b) **Tiles Connected Component** Find the size of the largest/smallest connected group of colored tiles.



(c) **Tiles Shortest Path** Find the length of the shortest path between the two marked tiles if it exists.



(d) **Missing Tiles** Find the correct shape and color of tiles to fill the blank region.



(e) **Tiles Composition** Choose the option that composes or decomposes into the top figure.

Figure 6. Examples of Topological and Tiling tasks.

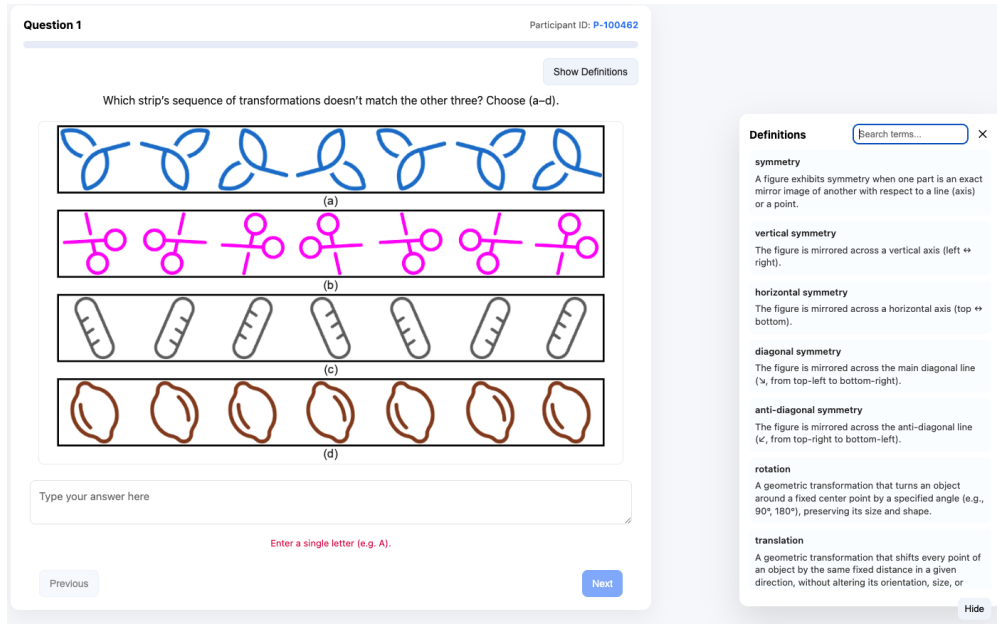


Figure 7. Web application interface used for the human evaluation. Participants were shown a visual prompt (image and/or text) and provided responses in the answer box.

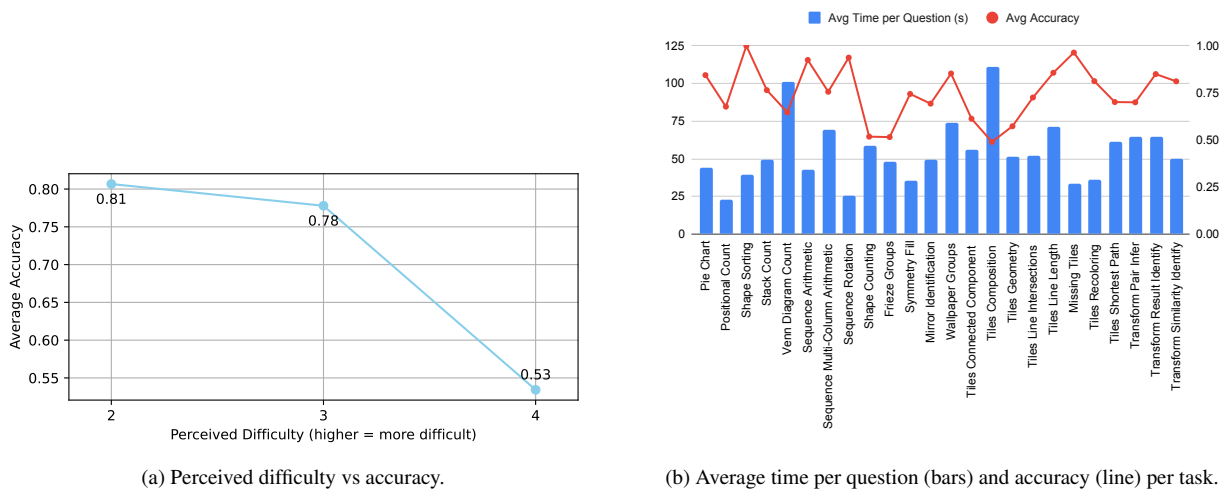


Figure 8. Human evaluation results. (a) Plot of participant perceived difficulty versus accuracy (b) Task-level time and accuracy.

the model gives a coherent explanation of how the large block can be decomposed into the three constituent parts, which aligns with the intended reasoning for the task.

The incorrect cases reveal different failure modes. In example (iii), the model appears to attempt a more sophisticated internal strategy for reasoning about the triangular tiling, although the explicit details do not appear in the response. In the end, it draws the wrong conclusion. In example (iv), the model again relies on counting the number of small squares in the tiles. All options and the three top blocks contain exactly 28 unit squares, yet the model incorrectly states that the top image contains 25 squares and that option (a) also contains 25. This suggests that

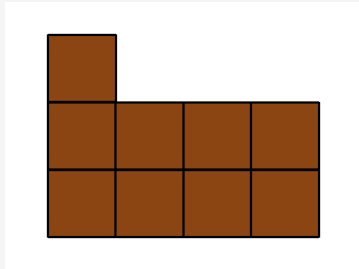
the model may be producing a post-hoc justification for an incorrect answer rather than performing a reliable visual count.

Overall, these examples show that GPT-5 has some genuine ability to reason about shape composition, but that its success depends heavily on extracting accurate low-level visual features, which remains an inconsistent step in its pipeline.

D.3. Tiles Line Length

Tiles Line Length is the task where GPT-5 performs the worst relative to humans, reaching only 14% accuracy compared to the human accuracy of 86.4%. Figure 11

(i)



Question: What is the square count in the figure?

Answer: 12

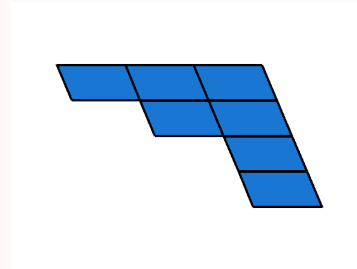
Response from GPT-5 (Correct):

Step-by-step:

- 1×1 squares: Count the unit squares. There are 4 in the bottom row, 4 in the middle row, and 1 on top = 9.
- 2×2 squares: These can only be formed within the 2×4 rectangle at the bottom. There are 3 such squares (columns 1–2, 2–3, and 3–4).
- 3×3 or larger: Not possible because the top row has only one square.

Total squares = $9 + 3 = 12$. 12

(ii)



Question: What is the number of parallelograms?

Answer: 19

Response from GPT-5 (Correct):

Step 1: Count 1×1 parallelograms (unit tiles). - There are 7 unit tiles.

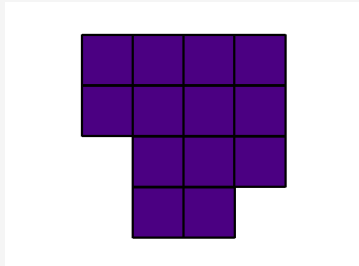
Step 2: Count wider shapes (height 1). - 2×1 : top row has 2, second row has 1 $\rightarrow 3$. - 3×1 : only on the top row $\rightarrow 1$.

Step 3: Count taller shapes (width 1). - 1×2 : along common columns: three in the rightmost column plus one in the middle column $\rightarrow 4$. - 1×3 : rightmost column, starting at rows 0 and 1 $\rightarrow 2$. - 1×4 : rightmost column across all rows $\rightarrow 1$.

Step 4: Count larger blocks. - 2×2 : only possible using rows 0–1 and columns 1–2 $\rightarrow 1$. - No bigger blocks fit. Total = $7 + 3 + 1 + 4 + 2 + 1 + 1 = 19$.

Final answer: 19

(iii)



Question: What is the square count in the figure?

Answer: 20

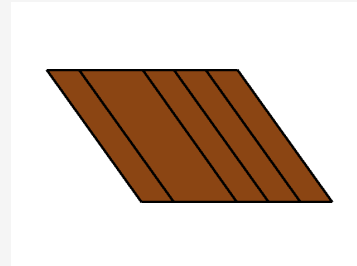
Response from GPT-5 (Incorrect):

Step-by-step count of all axis-aligned squares:

- 1×1 squares: Rows have 4, 4, 3, **3**, 2 unit squares $\rightarrow 4+4+3+3+2 = 16$.
- 2×2 squares: Between rows (1,2): 3 Between rows (2,3): **2** Between rows (3,4): **2** Between rows (4,5): 1 Total = $3+2+2+1 = 8$.
- 3×3 squares: Rows (1–3): 1 Rows (2–4): **1** Rows (3–5): **0** Total = 2.
- 4×4 or larger: none fit entirely in the figure.

Total number of squares = $16 + 8 + 2 =$ 26.

(iv)



Question: How many parallelograms are there in the figure?

Answer: 15

Response from GPT-5 (Incorrect):

Step-by-step reasoning:

- Inside the parallelogram there are **3** lines parallel to one pair of sides. Including the two outer sides, that makes **5** parallel lines in that direction.

- In the other direction, there are only the top and bottom sides (2 lines).

- Any parallelogram is formed by choosing two of the 5 parallel lines and using both the top and bottom sides.

Number of parallelograms = $C(5,2) \times C(2,2) = 10 \times 1 = 10$. 10

Figure 9. Example responses from GPT-5 on Shape Counting Task, with incorrect reasoning highlighted in red.

(i)

Answer: (a)
Response from GPT-5 (Correct):
 Step 1: Determine the areas of the three pieces on top. - L-hexamino: 6 unit squares - L-triomino: 3 unit squares - Domino: 2 unit squares Total area = 6 + 3 + 2 = 11.
 Step 2: Count areas of the candidates. - (a) has 11 squares. - (b) has 12 squares. - (c) has 10 squares. - (d) has 10 squares. Only (a) matches the required area of 11; hence (b), (c) and (d) cannot be tiled by the given pieces.
 Step 3: Feasibility check. Coloring the board in a checkerboard pattern: the hexomino and the domino cover equal numbers of each color, while the triomino covers one more of one color than the other, so the pieces together cover a region with black–white difference 1. Any region with 11 squares (odd area) has such a difference, so (a) is tileable (e.g., place the hexomino along the left and bottom forming an L, the triomino in the upper-left 2x2 corner, and the domino on the right protrusion). Therefore the only candidate that can be tiled by the top pieces is (a).

(ii)

Answer: (a)
Response from GPT-5 (Correct):
 Step-by-step reasoning:
 - The top figure is a 3x3 grid missing the top-left square (8 unit squares total).
 - It can be split into three parts: 1) An L-shaped triomino formed by the blue–yellow–magenta corner (top-right). 2) An L-shaped triomino formed by brown–orange–cyan at the lower-left. 3) A horizontal domino formed by orange–purple at the bottom-right.
 - Only bag (a) contains these exact three pieces (two L triominoes and one domino) with the matching color groupings. Other bags have the wrong shapes or number of squares.
 Final answer: (a)

(iii)

Answer: (a)
Response from GPT-5 (Incorrect):
 Reasoning (sketch):
 - On a triangular grid, color the unit triangles with three colors in the usual 3-coloring (so adjacent unit triangles get different colors). - For any polyiamond, the triple (number of color-1, color-2, color-3 triangles) is invariant under cutting and reassembling with rotations/reflections.
 - Counting the pieces in the “bag” gives one specific triple of counts; comparing these totals with the four candidate tiles, only tile (b) has the same color distribution. - Therefore the pieces can be assembled to form (b) and none of the other shapes.
 So the only possible connected tile is (b).

(iv)

Answer: (b)
Response from GPT-5 (Incorrect):
 Step 1: Count the area (number of unit squares) of the pieces in the bag.
 - Small piece (2-by-3 with a missing corner): 5 squares. - Plus-shaped piece with a small extension: 8 squares. - Large irregular piece: 12 squares. Total = 5 + 8 + 12 = 25 squares.
 Step 2: Count the area of each candidate tile. (a) 25 squares (b) 27 squares (c) 26 squares (d) 29 squares
 Only option (a) has the same total area as the three pieces, and the shapes can be fitted together (rotations/reflections allowed) to fill it without gaps or overlaps.
 Final answer: (a)

Figure 10. Example responses from GPT-5 on Tiles Composition Task, with incorrect reasoning highlighted in red.

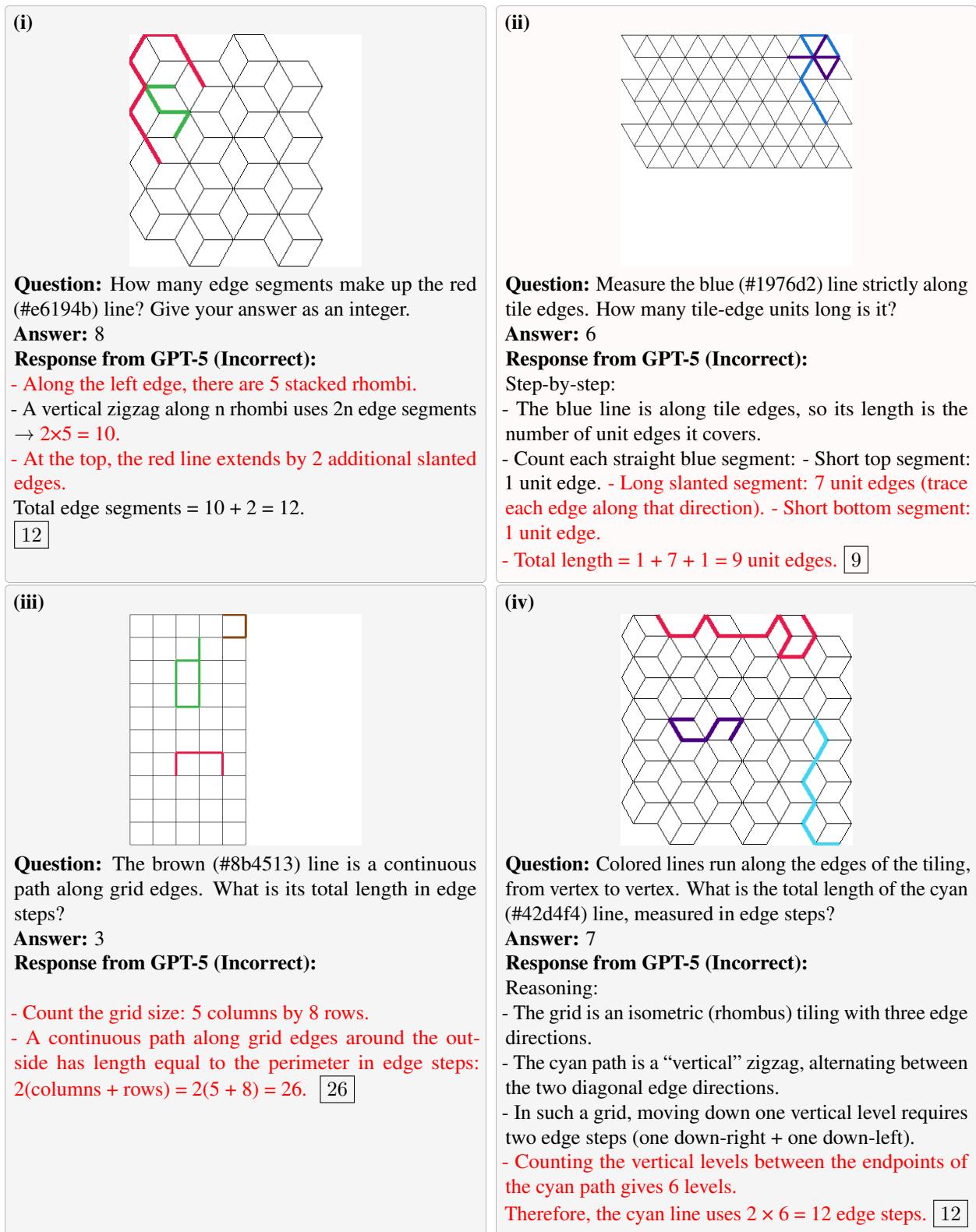


Figure 11. Example responses from GPT-5 on *Tiles Line Length* Task, with incorrect reasoning highlighted in red.

shows four representative examples in which the model produced incorrect responses. These cases illustrate sev-

eral different types of visual reasoning failures.

In example (i), the model incorrectly reports five

stacked rhombi rather than four and also assigns the wrong colors, stating that all segments are red. In example (ii), the model overcounts the length of the blue slanted segment. Part of this mistake may stem from conflating the blue and purple segments and treating them as a single continuous piece. In example (iii), the model appears to miss the brown segment entirely and instead attempts to compute the length of a continuous path along the outer boundary. This reasoning is also incorrect because the grid contains ten rows rather than eight, which invalidates the model’s length calculation. In example (iv), the model seems to infer six distinct vertical levels based on directional changes in the cyan segment, which again leads to an incorrect total.

Taken together, these examples suggest that GPT-5 relies on a collection of heuristics rather than a reliable visual understanding of the line structures. Humans can accurately isolate segments and identify endpoints, but the model does not consistently extract these visual primitives, which prevents it from solving the task correctly.

D.4. Tiles Recoloring

Tiles Recoloring is a challenging task for all models, including GPT-5, and Figure 12 shows four representative failure cases. Across these examples, GPT-5 appears to attempt several different strategies to solve the task, yet errors are introduced at various points in these strategies.

In example (i), the model misjudges the size of the yellow region and concludes that it has been shifted to the right in the second image. In reality, the yellow region is almost perfectly aligned between the two images, with only one cell recolored as magenta. This leads the model to output 12 rather than the correct answer of 1. In example (ii), GPT-5 attempts to convert both tiles into a letter-matrix representation with colors assigned to symbols. This is a valid approach, but the model makes errors when transcribing the right tile into matrix form, resulting in an incorrect comparison.

In example (iii), the model switches to a different algorithm. It identifies that some green cells in the left image have been recolored as cyan in the right image, and proposes counting the green cells in both images and taking their difference. However, it overcounts the green cells in both images by one, which leads to the wrong answer. In example (iv), the model fails to detect additional gray cells that appear only in the right image and therefore underestimates the number of mismatches.

Taken together, these examples indicate that GPT-5’s low accuracy on this task is driven primarily by failures in color-specific tile counting. Even when the model proposes valid high-level methods for solving the task, it struggles to carry them out reliably, because the core step of accurately identifying and counting colored cells across two tiles is error prone.

D.5. Frieze Groups

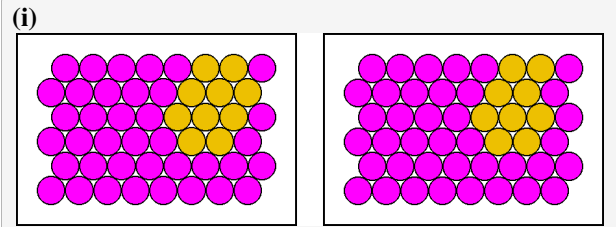
The Frieze Groups task has the lowest human accuracy (48.4%), and GPT-5 achieves only 26% accuracy. Since the task is presented in a multiple-choice format with four options, GPT-5’s performance is effectively at the level of random guessing. Figure 13 presents four representative cases of GPT-5 failure on the Frieze Groups task. The task requires identifying the symmetry pattern that governs the repetition of motifs along a one-dimensional strip, and the examples highlight several recurring challenges for the model.

In example (i), GPT-5 reasons that in strips (a), (c), and (d) the clocks differ by a 180° change of the clock hands. In reality, only strip (d) exhibits a 180° rotation, corresponding to group p2. Strips (a), (b), and (c) follow a horizontal reflection pattern consistent with the group p11m. The model, therefore, misclassifies the symmetry in multiple strips. In example (ii), the model makes several errors while analyzing the sequence. Strips (a), (b), and (c) follow a spinning sidle pattern (p2mg), which combines rotation with vertical reflection. Strip (d), in contrast, follows only a 180° rotation. GPT-5 incorrectly concludes that strip (c) repeats in the same orientation, although its visual change in orientation is more pronounced than in strip (d).

In example (iii), strips (a), (b), and (d) follow a combination of horizontal and vertical reflections (p2mm), while strip (c) follows a 180° rotation. The model incorrectly assigns the reflection to the strip (c) and fails to detect the horizontal reflection in the strips (a) and (b) and the vertical reflection present in strip (d). In example (iv), strips (b), (c) and (d) follow a reflection followed by a rotation (p2mg), while strip (a) uses a horizontal reflection (p11g). GPT-5 incorrectly infers that strips (b) and (d) remain upright and ignores the rotational component. It also incorrectly attributes a vertical reflection to strip (a) and, in strip (c), recognizes the rotation but fails to identify the reflection. These examples suggest that GPT-5 struggles with fine-grained symmetry analysis in one-dimensional sequences.

D.6. Wallpaper Groups

Human accuracy is much higher on the Wallpaper Groups task compared to the Frieze Groups task, reaching 86.4%, and GPT-5 also performs better at 48%. Figure 14 shows four representative GPT-5 responses, with two correct examples in the top row and two incorrect ones in the bottom row. In this task, the two-dimensional layout makes the global symmetry more visually apparent, which likely explains the higher human accuracy. GPT-5 is able to identify the correct symmetry pattern in some cases, but incorrect examples illustrate a recurring limitation. Although humans can easily integrate the global structure of a repeating 2D pattern, GPT-5 often focuses on local



Question: How many cells are different between the left and right images? Answer as an integer.

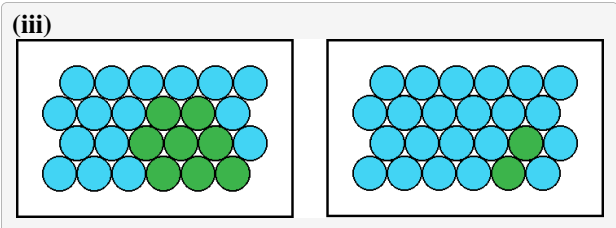
Answer: 1

Response from GPT-5 (Incorrect):

Step-by-step reasoning:

- The yellow region in each image forms a 3-by-4 rectangle (12 cells).
- In the right image this rectangle is shifted one cell down and one cell to the right relative to the left image.
- The overlap between a $w \times h$ rectangle and the same rectangle shifted by $(1,1)$ has size $(w-1)(h-1)$.
- Cells that differ = $2 \times [w \cdot h - (w-1)(h-1)] = 2(w + h - 1)$.
- With $w = 3$ and $h = 4$: $2(3 + 4 - 1) = 12$.

Final answer:



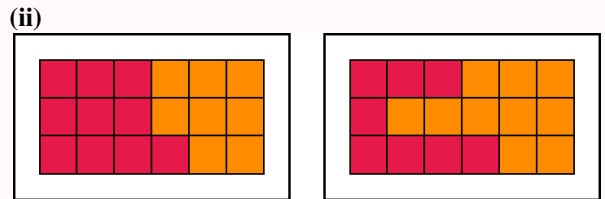
Question: A difference is any cell with unequal colors (including blank vs. filled). How many differences are there?

Answer: 6

Response from GPT-5 (Incorrect):

We compare the two grids cell by cell. Treat each possible circle position as a cell and count a difference whenever the two pictures have different colors at that position.

- There are 26 cells in each picture (6-7-6-7 in the four rows).
- On the left, 9 of those cells are green; on the right, only 1 cell is green.
- That one green cell on the right is also green on the left, so it is not a difference.
- Thus the number of cells with different colors is $9 + 1 - 2 \times 1 = 8$.



Question: Compare the two tiles. How many cells do not match between left and right? Give an integer.

Answer: 2 **Response from GPT-5 (Incorrect):**

Step-by-step reasoning:

- Each tile is a 3×6 grid.
- Left tile colors (R=red, O=orange):

Row1: R R R O O O

Row2: R R R O O O

Row3: R R R R O O

- Right tile colors:

Row1: R R O O O O

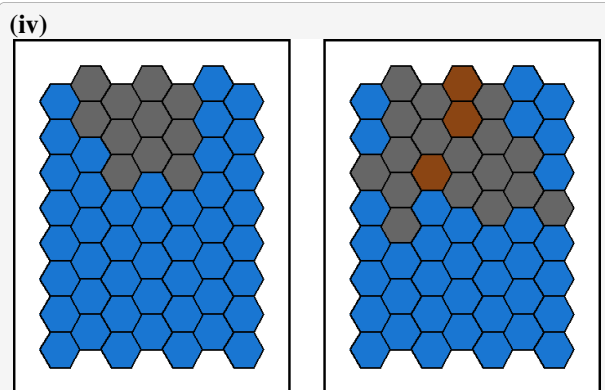
Row2: R O O O O O

Row3: R R R O O O

- Compare cell by cell. The cells that differ are: (row1,col3), (row2,col2), (row2,col3), (row3,col4).

- Total mismatches = 4.

Final answer:



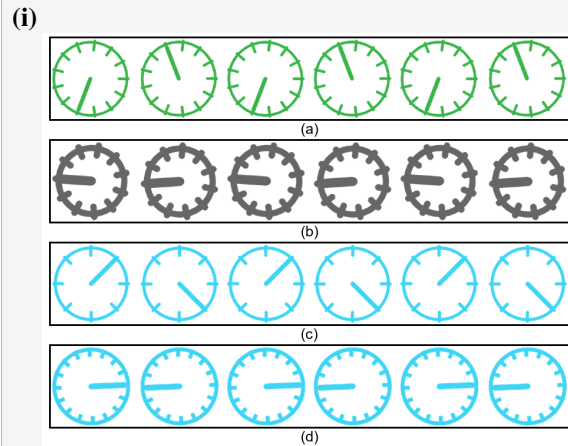
Question: Count cells whose colors do not match between left and right (filled vs. blank also counts). Provide an integer.

Answer: 12

Response from GPT-5 (Incorrect):

Only two cells differ. In the right image, two hexes are brown where the left image has grey; all other cells match.

Figure 12. Example responses from GPT-5 on Tiles Recoloring Task, with incorrect reasoning highlighted in red.



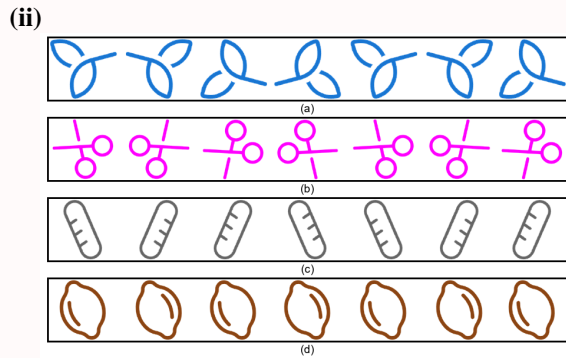
Question: Inspect how shapes relate to their neighbors. Which strip follows a different rule than the rest?

Answer: (d)

Response from GPT-5 (Incorrect):

(b)

Reason: In strips (a), (c) and (d), adjacent clocks are related by a 180° change of the hand (they alternate between two opposite orientations). In strip (b) all the hands point in the same direction, so it follows a different neighbor relation.



Question: Which strip's sequence of transformations doesn't match the other three? Choose (a–d).

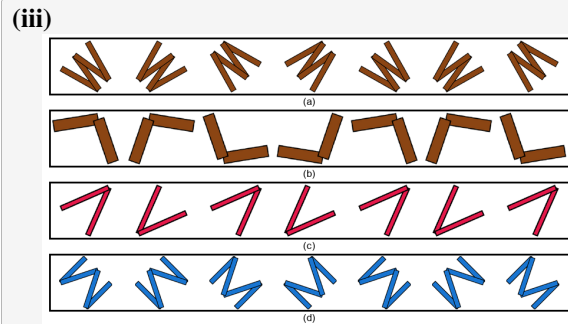
Answer: (d)

Response from GPT-5 (Incorrect):

Step-by-step reasoning:

- In strips (a), (b), and (d), each motif alternates between left-right mirror images of the previous one (reflection).
 - In strip (c), the motif repeats with the same orientation each time; there is no alternating reflection.
- Therefore, the sequence of transformations in (c) doesn't match the other three.

c



Question: Only one strip uses a different symmetry between adjacent motifs. Select it (a–d).

Answer: (c)

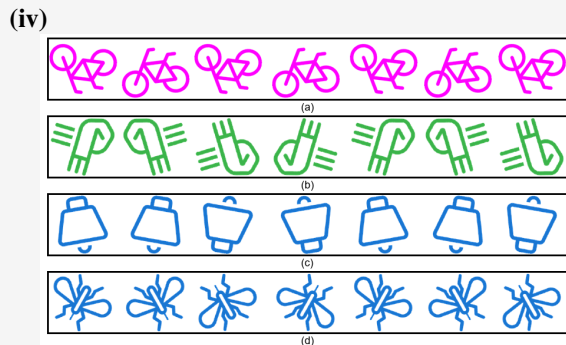
Response from GPT-5 (Incorrect):

Step-by-step reasoning:

- Compare each pair of neighboring motifs and identify the symmetry that maps one to the next.
- In strips (a), (b) and (c), adjacent motifs are mirror images across a vertical line between them (left-right reflection).
- In strip (d), adjacent motifs alternate as "M" and "W", i.e., they are mirror images across a horizontal line (top-bottom reflection), not a vertical one.

Therefore the only strip using a different symmetry is

d



Question: Only one strip uses a different symmetry between adjacent motifs. Select it (a–d).

Answer: (a)

Response from GPT-5 (Incorrect):

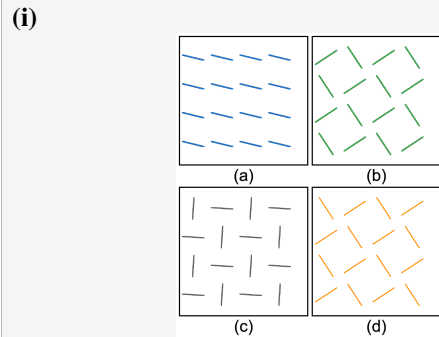
Step-by-step reasoning:

- Look at how one motif changes to the next.
- In (a), (b) and (d) the motifs remain upright from one to the next; they just switch left-right. That means adjacent motifs are related by a mirror reflection in a vertical line.
- In (c) the bell alternates upright and upside-down: the handle and clapper swap top and bottom. That relation is a 180° rotation (half-turn), not a mirror.

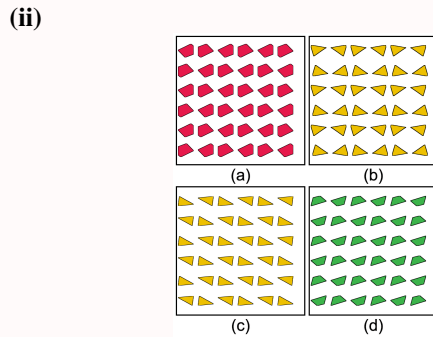
Therefore, only strip (c) uses a different symmetry between adjacent motifs.

c

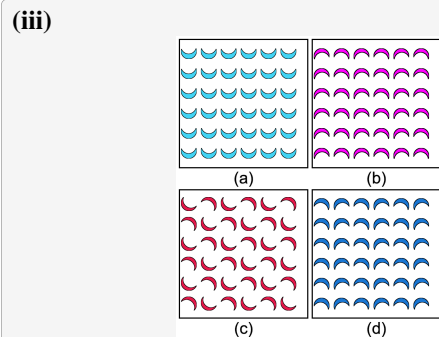
Figure 13. Example responses from GPT-5 on *Frieze Groups* task, with incorrect reasoning highlighted in red.



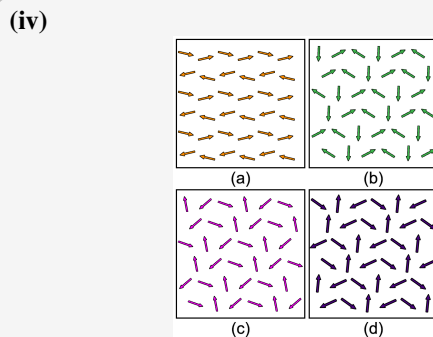
Answer: (a)
Response from GPT-5 (Correct):
 Step-by-step:
 - Look for rotational symmetries.
 - Panels (b), (c) and (d) each have points with 90° rotational (fourfold) symmetry: rotating the pattern by a quarter-turn about the centers of the squares maps the pattern onto itself. They belong to a p4-type wallpaper rule.
 - Panel (a) lacks any quarter-turn symmetry; it only has translations and 180° rotations (p2-type).
 Therefore, the panel that uses a different wallpaper rule is .



Answer: (b)
Response from GPT-5 (Correct):
 Step-by-step reasoning:
 - Look at how each motif relates to its nearest neighbors.
 - In panel (b), every immediate neighbor (left, right, up, and down) is the same motif rotated by 180°. The orientations therefore alternate in a checkerboard pattern; the symmetry is based on half-turns between each pair.
 - In panels (a), (c), and (d), motifs within a row keep the same orientation and the next row is obtained by a glide/mirror of the previous row (not by a 180° rotation between immediate neighbors).
 Thus, (b) follows a different wallpaper rule from the other three.



Answer: (c)
Response from GPT-5 (Incorrect):
 Step-by-step reasoning:
 - Look for symmetries other than translation (flips, rotations, glides) that generate each repetitive patch.
 - In (a), every crescent has the same orientation everywhere. The pattern can be produced by pure translations only; there are no flips/rotations/glides.
 - In (b), (c), and (d), neighboring motifs are related by half-turns or flips (and glides), so these patterns require additional symmetries beyond simple translations.
 Therefore the one generated by a different set of operations is patch (a).
 Final answer:



Answer: (a)
Response from GPT-5 (Incorrect):

 Reasoning:
 - In (a), (b), and (d) each arrow has a partner of opposite direction positioned so that 180° rotations (and corresponding glide reflections) map the pattern to itself.
 - In (c) almost all arrows point in the same NE direction; after a reflection or a 180° rotation they would point SW, which is absent. Thus (c) lacks those symmetries and is the odd one out.

Figure 14. Example responses from GPT-5 on *Wallpaper Groups* task, with incorrect reasoning highlighted in red.

neighborhoods and fails to reconcile those local transformations with the global symmetry of the wallpaper. This mismatch leads the model to select the wrong pattern even when the overall structure is clear in the image.

E. GPT-5 vs GPT-5 Mini

Stack Count. GPT-5 outperforms GPT-5 Mini by 36% on the Stack Count task. In this task, larger shapes of equal area are stacked on top of each other and smaller black shapes are placed on top of the stack. Figure 15 shows representative examples of the two models for two questions where GPT-5 gives the correct answer and GPT-5 Mini does not. GPT-5 is able to reason about the equal areas of the stacked shapes and infer the area of the occluded shape by comparing it with the visible one on top. GPT-5 Mini, while able to grasp the need to extrapolate the area of hidden layers, struggles with accurately counting the smaller shapes and therefore arrives at incorrect answers.

Transform Result Identify. In this task, the model must select the correct output image after a specified transformation is applied to the input motif. GPT-5 outperforms GPT-5 Mini by 17% on this task. Figure 16 shows two representative examples in which GPT-5 answers correctly and GPT-5 Mini answers incorrectly. GPT-5 appears to reason more reliably about the global shape and structure of the motif, allowing it to match the transformed result. In contrast, GPT-5 Mini struggles with these global cues and, in example (ii), also fails to recognize or describe the anti-diagonal symmetry present in the image, leading to an incorrect prediction.

Transform Similarity Identify. In this task, the model must identify which image is similar or dissimilar to the reference motif under Euclidean similarity, which includes uniform scaling together with the D4 group of rotations and mirror reflections. Each question contains one correct similar option or one correct dissimilar option. GPT-5 outperforms GPT-5 Mini in this task by 12%. Figure 17 shows example responses for two questions in which GPT-5 selects the correct answer and GPT-5 Mini does not. As in the Transform Result Identify task, GPT-5 demonstrates a stronger ability to reason about the global structure of the motif and how the transformation affects its overall form. Although GPT-5 occasionally reaches the correct answer with partially flawed reasoning, this is often sufficient in the multiple-choice format. GPT-5 Mini, on the other hand, tends to focus on local features and does not integrate them into a coherent global comparison, which leads to incorrect conclusions in these examples.

Sequence Arithmetic. GPT-5 Mini outperforms GPT-5 by 18% on the Sequence Arithmetic task, which requires identifying the missing entry in an arithmetic sequence of motifs. Figure 18 shows two examples where GPT-5 Mini answers correctly while GPT-5 does not. In both cases, GPT-5 makes multiple mistakes when counting the motifs in the grids and analyzing how their quantities change throughout the sequence. In example (i), GPT-5 focuses only on the bottom row of each grid and incorrectly counts the number of crescents in the first grid as 1 instead of 3, which leads to the wrong answer. In the second example, the model again miscounts, this time failing to identify the correct number of rings in each grid, and therefore selecting the incorrect option. GPT-5 Mini, on the contrary, appears to perform a more reliable count for this task.

Sequence Multi-Column Arithmetic. In the Sequence Multi-Column Arithmetic task, each grid contains several columns, and each column follows its own arithmetic progression across the sequence. In contrast to the Sequence Arithmetic task, GPT-5 outperforms GPT-5 Mini by 12%. Figure 19 shows two representative cases: in example (i), GPT-5 is correct while GPT-5 Mini is incorrect, and in example (ii), GPT-5 Mini answers correctly while GPT-5 does not. Interestingly, in both cases, the models display subtle reasoning errors even when selecting the correct answer. Both struggle to track the progression of individual columns and often miscount the motifs within each column. Their explanations reveal that they rely on shallow heuristics or partial observations rather than a full analysis of the multi-column structure.

Summary These results, together with the earlier discussion in Section D, reveal several key insights about the GPT-5 model family on the tasks in SPHINX. Both models struggle with counting over abstract symbols, which is evident in motif-based tasks such as *Sequence Arithmetic* and *Sequence Multi-Column Arithmetic*, as well as tile-based tasks such as *Tiles Line Length* and *Tiles Recoloring*. This difficulty becomes even more pronounced when counting interacts with additional structure, as in the *Sequence Multi-Column Arithmetic* task. Although models often produce coherent textual reasoning, errors introduced during the visual counting stage frequently lead to incorrect answers.

For tasks involving symmetry and transformation, GPT-5 is generally better equipped than GPT-5 Mini to reason about the global structure of the image, although both lag behind human performance, as seen in the *Wallpaper Groups* task. The strong performance of GPT-5 on *Shape Counting* also suggests that the model may surpass humans in settings that demand substantial mental arithmetic and relatively less visual extraction.

Overall, these findings show that the success of GPT-5 strongly depends on whether the task requires accurate visual extraction or primarily symbolic reasoning, with its largest failures driven by perceptual errors rather than conceptual ones.

F. Discussion of RLVR trained models

Training Dynamics. Figure 20 shows the evolution of the training reward over the course of the RL training for all four models, and Figure 21 shows the corresponding average response length measured on the training set. For the Qwen3-VL models, the average response length initially decreases and then gradually increases as training progresses, while for the Qwen2.5-VL models, it remains much closer to that of the base model throughout training. The reward curves also indicate that the models are not fully saturated and could benefit from additional training, which remains a promising direction for future work.

Response Length and Accuracy. Figure 22 shows the average response length and accuracy across the SPHINX benchmark tasks for the four RLVR-trained models. The Qwen3-VL models, both base and RLVR-trained, consistently produce longer responses than the Qwen2.5-VL models across all tasks.

Tasks 4 (Pie Chart), 2 (Shape Sorting), and 6 (Venn Diagram) are among the shortest tasks that still achieve high accuracy. Although the Pie Chart task already showed strong performance before training, RLVR training led to clear improvements on the other two tasks. One possible explanation is that these tasks do not rely on Motifs or Tiles, which limits the visual variety of the problem instances and results in test data that are not substantially different from the training data.

In contrast, Task 23 (Tiles Shortest Path) has the longest average description for both Qwen2.5-VL models and yields below 20% accuracy, with only slightly better performance from the Qwen3-VL models. Tasks 25 (Tiles Composition), 12 (Symmetry Fill), 17 (Transform Similarity Identify), and 13 (Frieze Groups) also show long response lengths for the Qwen3-VL models while remaining below 40% accuracy. Tiles Composition is not included in the training set, but the other three tasks are, suggesting that these tasks are inherently difficult for the models to learn even with RLVR training.

Example Response. We provide several example responses from the base models and the RLVR-trained models in Figure 23, Figure 24, Figure 25 and Figure 26.

G. Evaluation

G.1. Prompt

```
Solve the problem by reasoning step
by step. Then provide the final
answer inside \boxed{}
```

```
Question: {}
```

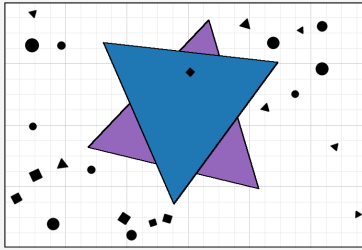
G.2. Evaluation Protocol

We evaluated all model outputs using a two-stage extraction and scoring pipeline. For each question, we first infer the answer type from the ground-truth label (multiple choice, integer, or free-form text) and then apply a set of heuristic rules to extract a canonical answer from the model’s raw response. When this heuristic extraction succeeds, the answer is scored directly using `mathruler` with strict type-consistent normalization. If the heuristic stage cannot produce a valid extraction, we query a small GPT-5-mini model at temperature 0 with a short few-shot prompt to perform answer extraction. The extracted output is then compared to the ground truth.

H. Task Accuracy

Tables 1a and 1b present the task-level accuracies of the evaluated models on the 25 tasks in the SPHINX benchmark.

(i)



Question: Equal-area colored sheets are stacked first, and the small black squares are added on top. How many small black squares are strictly inside the purple triangle?

Answer: 1

Response from GPT-5 (Correct):

- The black squares are on top of the stacked colored triangles.
- We count only those lying strictly inside the purple triangle's interior (not on edges).
- Inspecting the figure, only the small black diamond near the center lies inside the purple triangle; all other black squares are outside it.

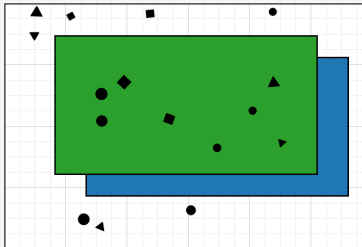
1

Response from GPT-5 Mini (Incorrect):

Count the little rotated black squares that lie strictly within the purple triangle (not on an edge). **Marking them one-by-one on the image gives four such squares.**

Final answer: 4

(ii)



Question: The equal-area sheets are laid down first and may occlude each other; the black circles are added afterwards on top. How many small black circles are strictly inside the boundary of the blue rectangle?

Answer: 4

Response from GPT-5 (Correct):

Step-by-step reasoning:

- The blue and green rectangles are equal in size; the blue is shifted down and to the right under the green.
- Therefore, the unseen left and top edges of the blue rectangle lie inside the green rectangle by the same amounts as the blue overhang to the right and to the bottom.
- A point (circle) is inside the blue rectangle iff it is to the right of that internal vertical line and below that internal horizontal line.
- Checking the circles, exactly four of the small black circles satisfy both conditions.

Final answer: 4

Response from GPT-5 Mini (Incorrect):

Step 1 — locate the true blue rectangle (including the parts hidden under the green sheet).

- The visible blue strip to the right and bottom shows the blue rectangle's full horizontal and vertical extents; those edges continue straight under the green sheet.

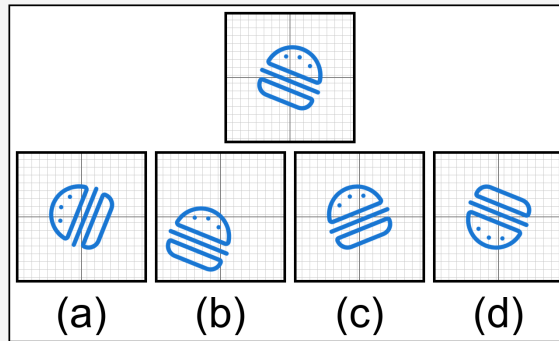
Step 2 — find which black circle centers lie strictly inside those edges.

- **Using the blue rectangle edges inferred from the visible blue border, there are three small black circles whose centers lie strictly inside the blue rectangle (one on the exposed lower/right blue area and two lying on top of the portion of the blue sheet that is hidden under the green).**

Final answer: 3

Figure 15. GPT-5 vs. GPT-5 Mini response on *Stack Count* task, with incorrect reasoning highlighted in red.

(i)



Question: After rotate 90° counterclockwise is applied to the top shape, which option (a)–(d) matches the outcome?

Answer: (a)

Response from GPT-5 (Correct):

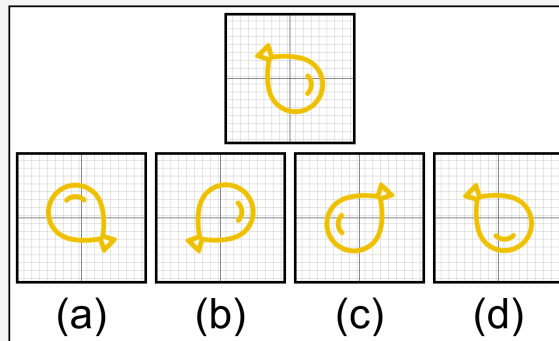
Step-by-step: - A 90° counterclockwise rotation moves the top semicircular bun to the left side of the figure.
- The two thin layers and the bottom bun, originally below the top bun, become vertical elements to the right of the semicircle.
Only option (a) shows the bun on the left with the three parallel layers to its right.

Response from GPT-5 Mini (Incorrect):

Rotate 90° CCW moves the features on the right of the original up and flips the stripes' slope. **Option (c) is the only choice where the three dots have moved to the top of the bun and the stripes have the correct rotated orientation.**

Answer:

(ii)



Question: Look at the top figure. If you apply reflect across the anti-diagonal (\swarrow), which option (a)–(d) is the correct result?

Answer: (a)

Response from GPT-5 (Correct):

Step-by-step reasoning:
- The anti-diagonal \swarrow is the line $y = x$ (from top-right to bottom-left).
- Reflecting across this line swaps x and y : $(x, y) \rightarrow (y, x)$.
- So features move: the right-side mark goes to the top, and the top-left arrow moves to the bottom-right.
- Among the options, only (a) shows the mark on top and the arrow at bottom-right.

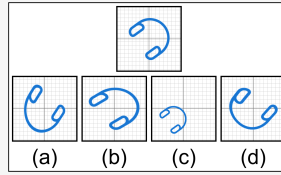
Response from GPT-5 Mini (Incorrect):

Reflecting across the anti-diagonal (the line $y = -x$) sends each point (x,y) to $(-y,-x)$. Applying that to the top figure produces the orientation and placement shown in option (d).

Final answer:

Figure 16. GPT-5 vs. GPT-5 Mini response on *Transform Result Identify* task, with incorrect reasoning highlighted in red.

(i)



Question: Which option (a)–(d) is similar to the top figure under uniform scaling and rigid/mirror motions?

Answer: (c)

Response from GPT-5 (Correct):

Step-by-step reasoning: - The original figure is a C-shaped arc with two small pill-like loops on the inside, one near each end, with the same relative tilt. - Similarity allows only uniform scaling, rotation, reflection, and translation; thus the curvature of the main arc and the relative positions/tilts of the two small loops must match. Check the options: - (a) and (d): one of the small loops has the opposite tilt relative to the arc, so they cannot be obtained by rigid motion/scale. - (b): the spacing and curvature are different (the arc is “stretched” and the loops sit at different relative distances), so not a uniform scale of the original. - (c): matches the curvature and the placement/tilts of both loops; it is simply a scaled copy.

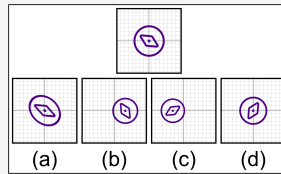
Final answer: (c)

Response from GPT-5 Mini (Incorrect):

Look at three invariant relations that similarity (with rotation/translation/reflection allowed) must preserve: (1) the two small rounded rectangles lie on the same circular arc, (2) their relative angular positions around the arc (one near the upper-left, the other near the lower-right) and (3) the direction each small rectangle is tilted (both are tilted the same way along the arc). Only option (b) preserves all three at once after a rotation/scale — the other choices place one or both small rounded rectangles in the wrong location or with the wrong tilt relative to the arc.

Therefore the similar figure is (b). (b)

(ii)



Question: Choose the option (a)–(d) that is not obtainable by uniform scaling of the top plus rotation/reflection/translation.

Answer: (a)

Response from GPT-5 (Correct):

Step-by-step reasoning:
 - Allowed operations: uniform scaling + rotation/reflection/translation. These are similarity transformations, which preserve shapes and map circles to circles (no squashing).
 - Inspect options: (b), (c), and (d) all keep the outer boundary circular and preserve proportions, so they can come from the top via scaling/rotation/translation.
 - Option (a) shows the outer “circle” squashed into an ellipse, with the inner parts distorted similarly. This would require non-uniform scaling/shear, which is not allowed.

Final answer: (a)

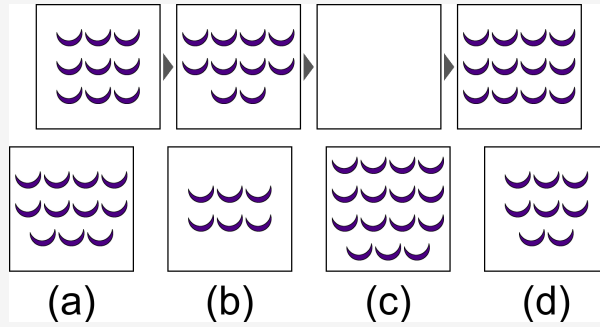
Response from GPT-5 Mini (Incorrect):

Key invariant: uniform scaling + rotation/reflection + translation preserves the relative position of features inside the figure. In particular, the vector from the circle center to the little dot (the diamond’s center) scales by the same factor as the circle radius, so the offset as a fraction of the circle radius cannot change. In the top picture the diamond’s center (the small dot) is clearly offset from the circle center. Any allowed transformation must keep a nonzero fractional offset. Option (b) shows the diamond exactly concentric with the circle (offset = 0), which cannot result from the top figure by the allowed operations.

Therefore the impossible option is (b).

Figure 17. GPT-5 vs. GPT-5 Mini response on *Transform Similarity Identify* task, with incorrect reasoning highlighted in red.

(i)



Question: Which option (a)–(d) below should replace the blank to complete the pattern shown in the top row?

Answer: (a)

Response from GPT-5 (Incorrect):

The top two rows remain full in every panel; only the bottom row changes.

From panel 1 to 2, two crescents disappear from the bottom-right. To return to a full grid in panel 4, one crescent is added back in each step.

So the missing third panel should have three crescents in the bottom row (still missing the last one).

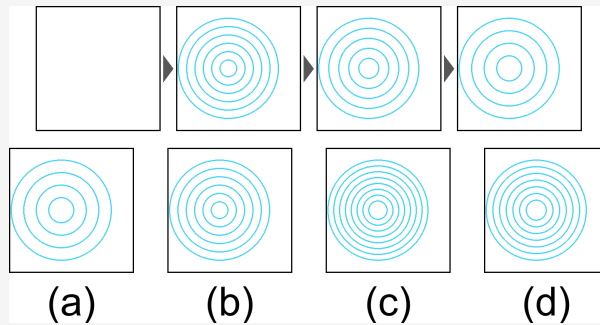
Therefore the correct option is c.

Response from GPT-5 Mini (Correct):

Count the crescents in the top row: left = 9, next = 10, final = 12. The sequence must increase by one each step, so the blank should contain 11 crescents. Option (a) shows 11 crescents.

Answer: (a)

(ii)



Question: Which option (a)–(d) below should replace the blank to complete the pattern shown in the top row?

Answer: (d)

Response from GPT-5 (Incorrect):

Reasoning:

- From the second to the fourth square, the concentric circles expand outward by a fixed amount each step.

- As they grow, the outermost circle moves out of the frame, so the number of visible circles decreases by one: $5 \rightarrow 4 \rightarrow 3$.

- Therefore, the square before the second one should show one more circle than the second: 6 circles, centered and evenly spaced.

The only option that matches this is option (b).

Final answer: b

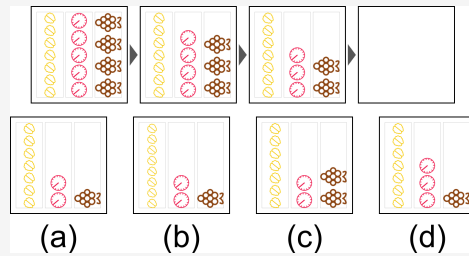
Response from GPT-5 Mini (Correct):

Each step from left to right removes one concentric ring. The first shown image (top row, second position) has 7 circles (center + 6 rings), then 6, then 5. So the missing leftmost image must have one more circle (center + 7 rings = 8 circles). Choice (d) is the only option with that many rings.

Final answer: d

Figure 18. GPT-5 vs. GPT-5 Mini response on *Sequence Arithmetic* task, with incorrect reasoning highlighted in red.

(i)



Question: Which panel (a)–(d) completes the sequence shown in the top row?

Answer: (a)

Response from GPT-5 (Correct):

Step-by-step reasoning:

- In each frame the whole pattern shifts to the right.
- As a column reaches the rightmost narrow area, only the bottom part remains visible: the number of visible objects in that column decreases by one each step.
- From the first to the third frame the pink column goes from 4 to 3 items visible, and the brown column from 4 to 2 visible.
- The next step should therefore show 2 pink dials and only 1 brown cluster, both still aligned at the bottom-right.

Only panel (a) matches this continuation.

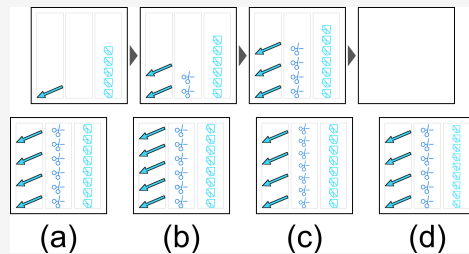
Final answer:

Response from GPT-5 Mini (Incorrect):

Answer:

Reason: The yellow column stays the same. Each step the middle (pink clocks) and right (brown clusters) lose one item and shift upward. **After three steps the middle has 2 clocks and the right has 1 cluster in the top-shifted positions — matching panel (b).**

(ii)



Question: Which panel (a)–(d) completes the sequence shown in the top row?

Answer: (a)

Response from GPT-5 (Incorrect):

Step-by-step:

- In each panel, the three columns contain arrows, circle-stick figures, and cubes respectively.
- The counts increase by one from panel to panel and also from left to right within a panel:
- Panel 1: 1 arrow, 2 circles, 3 cubes.
- Panel 2: 2 arrows, 3 circles, 4 cubes.
- Panel 3: 3 arrows, 4 circles, 5 cubes.
- **Therefore, the fourth should have 4 arrows, 5 circles, and 6 cubes.**

Only option (c) matches this.

Response from GPT-5 Mini (Correct):

Answer: (a)

Reason: Each step adds one more arrow in the left column and **one more scissors** in the middle column (scissors = arrows - 1), while **the right column of five envelopes stays constant**. After 1, 2, 3 arrows in the first three panels the fourth must show 4 arrows and 3 scissors — option (a).

Figure 19. GPT-5 vs. GPT-5 Mini response on *Sequence Multi-Column Arithmetic* task, with incorrect reasoning highlighted in red.

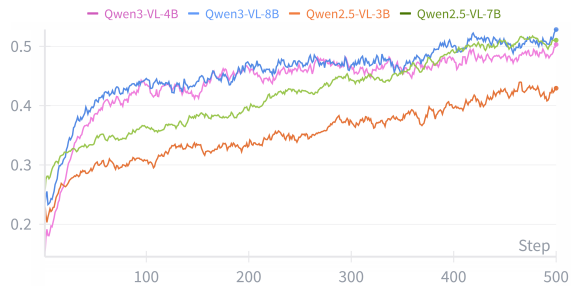


Figure 20. Training reward curves for the four models during RLVR training.

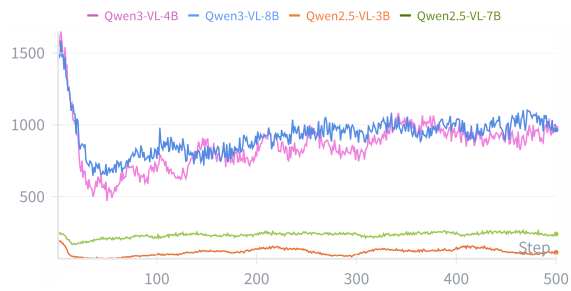


Figure 21. Average response length on the training set during RLVR training.

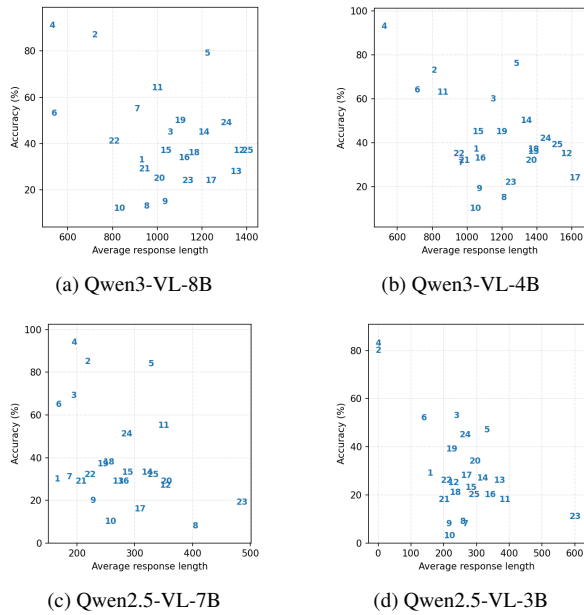
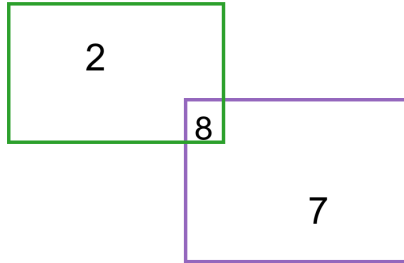


Figure 22. Average response length and accuracy for each task on the SPHINX benchmark for the four RLVR-trained models. Each point corresponds to a task ID and shows the model's average response length and accuracy for that task.



Question: What is the total of the numbers present simultaneously in the purple and green rectangle(s)?

Answer: 8

Response from Qwen3-VL-4B (Incorrect):

We are given two rectangles:

- A **green rectangle** containing the number **2**.
- A **purple rectangle** containing the number **8** and **7**.

The question asks: **What is the total of the numbers present simultaneously in the purple and green rectangle(s)?**

Step 1: Understand "simultaneously in the purple and green rectangle(s)"

This means we are looking for numbers that appear **in both** rectangles.

But note: the green rectangle has only the number 2. The purple rectangle has the numbers 8 and 7. There is no overlap between the numbers in the green rectangle and the numbers in the purple rectangle. So, the set of numbers common to both rectangles is empty.

Step 2: Compute the total

The total of an empty set is **0**.

Final Answer:

Response from Qwen3-VL-4B-RL (correct):

Answer:

We are given two overlapping rectangles:

- A **green rectangle** containing the number **2**.
- A **purple rectangle** containing the number **7** and overlapping with the green rectangle at a point where the number **8** is located.

The question asks: **What is the total of the numbers present simultaneously in the purple and green rectangle(s)?**

— ### Step-by-step reasoning:

1. The green rectangle has the number **2**.
2. The purple rectangle has the number **7**.
3. The two rectangles overlap, and in the overlapping region, there is the number **8**.

The key phrase is: **simultaneously in the purple and green rectangle(s)** — meaning, numbers that are **in both** rectangles.

The number **8** is located **at the intersection** of the two rectangles. Therefore, it is present in **both** the green and purple rectangles.

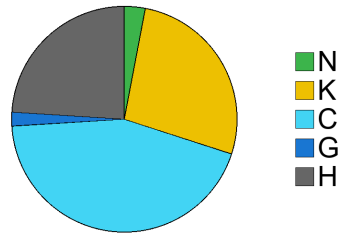
The numbers **2** and **7** are only in their respective rectangles — **2** is only in the green rectangle, and **7** is only in the purple rectangle. So they are not present in both.

Thus, the only number that is present simultaneously in both rectangles is **8**.

Final Answer:

The total of the numbers present simultaneously in both rectangles is just **8**, since it's the only number shared between them.

Figure 23. Qwen3-VL-4B vs. Qwen3-VL-4B-RL response on *Venn Diagram* task, with incorrect reasoning highlighted in red.



Question: Select the option (a)–(d) that lists categories in decreasing share (largest → smallest).

(a) C,K,H,N,G (b) C,N,H,K,G (c) C,H,K,N,G (d) C,G,H,N,K

Answer with one letter (a–d).

Answer: (a)

Response from Qwen3-VL-8B (Incorrect):

To solve this, we need to determine the relative sizes of the slices in the pie chart and then order the categories from largest to smallest.

Looking at the pie chart:

- The **light blue** slice (C) is the **largest** portion — it takes up more than half the circle. - **The dark grey slice (H) is the second largest.** - **The yellow slice (K) is the third largest** - The **green** slice (N) is a small slice, but larger than the blue slice (G). - The **blue** slice (G) is the **smallest**.

So, ordering from largest to smallest:

1. C (largest) 2. H 3. K 4. N 5. G (smallest)

This matches option **(c)**: C, H, K, N, G

Let's verify the other options quickly:

- (a) C,K,H,N,G → K before H? No, H is larger than K.
- (b) C,N,H,K,G → N before H? No, H is larger than N.
- (d) C,G,H,N,K → G before H? No, G is smallest, H is second largest.

Only option (c) has the correct order.

Final Answer:

Response from Qwen3-VL-8B-RL (correct):

To determine the correct option, we need to analyze the pie chart and rank the categories by their share from largest to smallest.

Looking at the pie chart:

- The **light blue** slice (C) is the **largest** portion. - The **yellow** slice (K) is the **second largest**. - The **gray** slice (H) is the **third largest**. - The **green** slice (N) is the **fourth largest**. - The **dark blue** slice (G) is the **smallest**.

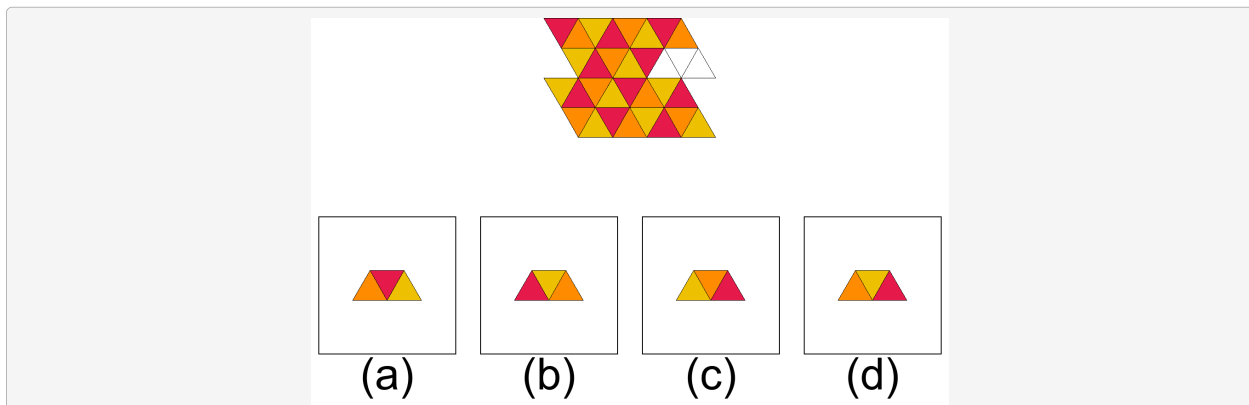
So, the order from largest to smallest is: **C > K > H > N > G**

Now, let's check the options:

- (a) C,K,H,N,G → This matches our order exactly.
- (b) C,N,H,K,G → Incorrect, because N is smaller than H and K.
- (c) C,H,K,N,G → Incorrect, because H is smaller than K.
- (d) C,G,H,N,K → Incorrect, because G is the smallest, not second.

Therefore, the correct answer is **(a)**.

Figure 24. Qwen3-VL-8B vs. Qwen3-VL-8B-RL response on *Pie Chart* task, with incorrect reasoning highlighted in red.



Question: The shape and position of the blank region are fixed in the top figure. Which option (a)–(d) restores its correct coloring?

Answer: (d)

Response from Qwen2.5-VL-3B (Incorrect):

To determine which option restores the correct coloring of the blank region in the top figure, we need to analyze the pattern and the given options step by step.

1. **Identify the Pattern:** - The top figure shows a pattern where the triangles are colored in a specific sequence. - The sequence of colors is: red, yellow, orange, red, yellow, orange, and so on.

2. **Analyze the Options:** - **Option (a):** This option shows a red triangle on the left and a yellow triangle on the right. - **Option (b):** This option shows a yellow triangle on the left and an orange triangle on the right. - **Option (c):** This option shows an orange triangle on the left and a red triangle on the right. - **Option (d):** This option shows a yellow triangle on the left and a red triangle on the right.

3. **Match the Pattern:** - The pattern in the top figure is red, yellow, orange, red, yellow, orange, and so on. - We need to find the correct sequence for the blank region.

4. **Determine the Correct Sequence:** - The blank region in the top figure is the third triangle from the left in the second row. - According to the pattern, the third triangle in the second row should be orange.

5. **Conclusion:** - The correct option that matches the pattern is option (b), which shows an orange triangle on the left and a yellow triangle on the right.

Therefore, the correct answer is b.

Response from Qwen2.5-VL-3B-RL (correct):

To determine the correct coloring for the blank region, we need to analyze the pattern of colors in the given figure and apply it to the blank region.

1. **Identify the pattern in the given figure:** - The figure is composed of smaller triangles. - The colors of the triangles alternate in a specific pattern.

2. **Analyze the colors in the given figure:** - The top-left triangle is red. - The triangle to its right is yellow. - The triangle below the red triangle is orange. - The triangle to the right of the orange triangle is red. - This pattern continues throughout the figure.

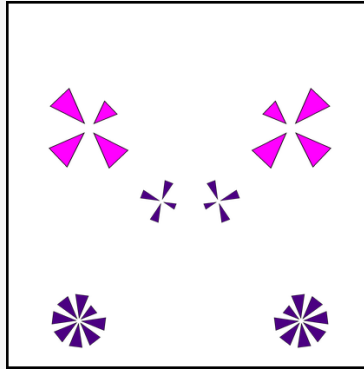
3. **Determine the color of the blank region:** - The blank region is located in the middle of the figure. - The pattern suggests that the blank region should be the same color as the triangle directly above it.

4. **Identify the color of the triangle directly above the blank region:** - The triangle directly above the blank region is yellow.

5. **Conclusion:** - The blank region should be colored yellow to maintain the pattern.

Therefore, the correct answer is d.

Figure 25. Qwen2.5-VL-3B vs. Qwen2.5-VL-3B-RL response on *Missing Tiles* task, with incorrect reasoning highlighted in red.



Question: Exactly one option describes the figure's mirror symmetry. Which is it? (a) reflection across a vertical line, (b) line symmetry along the anti-diagonal, (c) vertical + horizontal symmetry, (d) main-diagonal (\searrow) mirror, (e) horizontal line symmetry, (f) no reflection symmetry

Answer: (a)

Response from Qwen2.5-VL-7B (Incorrect):

To determine the type of symmetry in the given figure, let's analyze it step by step:

1. **Vertical Symmetry**: Check if the figure can be divided into two identical halves by a vertical line. - **The figure does not have a vertical line of symmetry because the shapes on the left and right sides are not mirror images of each other.**
2. **Horizontal Symmetry**: Check if the figure can be divided into two identical halves by a horizontal line. - The figure does not have a horizontal line of symmetry because the shapes above and below the horizontal line are not mirror images of each other.
3. **Anti-Diagonal Symmetry**: Check if the figure can be divided into two identical halves by a diagonal line from the top left to the bottom right. - The figure does not have anti-diagonal symmetry because the shapes on the left and right sides are not mirror images of each other along this diagonal.
4. **Main-Diagonal (\searrow) Symmetry**: Check if the figure can be divided into two identical halves by a diagonal line from the top right to the bottom left. - The figure does not have main-diagonal symmetry because the shapes on the top and bottom are not mirror images of each other along this diagonal.
5. **No Reflection Symmetry**: Check if the figure has any reflection symmetry. - **The figure does not have any reflection symmetry because none of the lines of symmetry mentioned above apply.**

Given the analysis, the figure does not have any of the described symmetries.

The final answer is

Response from Qwen2.5-VL-7B-RL (correct):

To determine the type of symmetry in the figure, let's analyze it step by step:

1. **Vertical Symmetry (a) and (c)**: - The figure appears to have vertical symmetry along the vertical axis passing through the center. - The shapes on the left and right sides of the vertical axis are mirror images of each other.
2. **Horizontal Symmetry (e)**: - The figure does not have horizontal symmetry. The shapes on the top and bottom do not mirror each other.
3. **Anti-Diagonal Symmetry (b)**: - The figure does not have anti-diagonal symmetry. The shapes do not mirror each other along the anti-diagonal line.
4. **Main-Diagonal (\searrow) Symmetry (d)**: - The figure does not have main-diagonal symmetry. The shapes do not mirror each other along the main-diagonal line.
5. **No Reflection Symmetry (f)**: - The figure does have some form of reflection symmetry, so this option is incorrect.

Given the analysis, the figure has vertical symmetry along the vertical axis passing through the center. Therefore, the correct answer is (a) reflection across a vertical line.

Figure 26. Qwen2.5-VL-7B vs. Qwen2.5-VL-7B-RL response on *Mirror Identification* task, incorrect reasoning highlighted in red.

Table 1. Task-level accuracy comparison across the 25 SPHINX tasks. All values are percentages.

Task Name	Hum	GPT-5	GPT-5 Mini	GPT-5 Nano	Gemini3 Flash	Claude4.5 Opus	Claude4.5 Sonnet
Positional Count	72.4	43	34	14	33	33	59
Shape Sorting	100.0	78	85	61	29	83	72
Stack Count	80.0	60	24	10	0	45	35
Pie Chart	88.0	92	91	77	66	77	93
Chart Comparison	96.8	97	96	60	22	51	81
Venn Diagram	65.0	52	57	53	51	62	62
Shape Counting	55.2	76	61	46	61	67	67
Tiles Line Length	86.4	14	18	10	14	16	11
Tiles Line Inters.	75.9	26	25	10	1	18	19
Tiles Recoloring	69.6	15	20	5	26	14	42
Mirror Identification	68.6	65	61	63	40	59	11
Symmetry Fill	76.0	61	54	30	11	41	40
Frieze Groups	48.4	26	32	29	21	24	34
Wallpaper Groups	86.4	48	51	41	32	53	52
Transform Res. Identify	87.9	68	51	37	31	43	37
Transform Pair Infer	63.2	57	51	26	17	30	25
Transform Sim. Identify	82.9	38	26	17	11	27	19
Sequence Rotation	95.2	55	50	35	17	29	37
Sequence Arithmetic	93.8	50	68	48	24	42	69
Sequence Multi-Column	74.3	41	29	28	14	22	37
Tiles Geometry	61.9	44	41	26	31	33	41
Tiles Conn. Component	57.1	44	41	22	29	35	56
Tiles Shortest Path	64.9	32	27	17	1	26	33
Missing Tiles	95.2	47	36	31	7	30	42
Tiles Composition	50.0	48	49	33	1	38	41

(a) Human and proprietary LVLMs.

Task Name	Hum	Intern VL3 8B	Intern VL3 38B	Llama 3.2 11B	Qwen 2.5-VL 3B	Qwen 2.5-VL 7B	Qwen 2.5-VL 32B	Qwen 3-VL 4B	Qwen 3-VL 8B	Qwen 3-VL 30B	Qwen 3-VL 235B	Molmo 2 4B	Molmo 2 8B	Molmo 2-O 7B
Positional Count	72.4	19	26	14	15	14	36	31	23	26	37	18	30	20
Shape Sorting	100.0	51	69	32	56	61	63	41	53	42	91	41	4	0
Stack Count	80.0	21	30	13	6	13	20	25	27	29	45	31	24	39
Pie Chart	88.0	50	79	30	54	74	91	69	78	83	97	40	2	0
Chart Comparison	96.8	26	23	23	24	29	52	46	43	57	64	23	29	20
Venn Diagram	65.0	31	44	15	17	31	45	35	31	50	48	33	30	32
Shape Counting	55.2	9	9	3	3	17	31	27	32	40	55	6	11	3
Tiles Line Length	86.4	8	10	9	3	4	8	8	8	9	9	7	8	7
Tiles Line Inters.	75.9	12	14	14	9	13	15	13	14	12	15	13	17	14
Tiles Recoloring	69.6	11	17	5	9	8	10	10	15	21	19	8	4	8
Mirror Identification	68.6	31	39	18	20	34	42	54	46	47	54	39	30	33
Symmetry Fill	76.0	22	30	24	24	26	25	27	32	24	34	17	17	26
Frieze Groups	48.4	34	30	22	24	27	34	26	20	28	40	21	23	30
Wallpaper Groups	86.4	33	36	21	23	28	33	38	39	37	45	19	23	25
Transform Res. Identify	87.9	26	27	27	21	26	32	37	43	41	37	24	28	30
Transform Pair Infer	63.2	18	26	22	19	15	18	14	35	25	20	15	18	16
Transform Sim. Identify	82.9	23	25	26	23	16	19	29	24	24	24	21	26	16
Sequence Rotation	95.2	30	23	24	18	24	26	32	32	22	45	25	20	27
Sequence Arithmetic	93.8	31	31	23	27	33	43	44	38	38	56	24	24	18
Sequence Multi-Column	74.3	15	26	20	25	27	22	23	26	23	29	18	21	21
Tiles Geometry	61.9	22	24	10	10	16	34	19	23	20	36	18	17	17
Tiles Conn. Component	57.1	28	29	16	19	16	28	38	29	29	45	28	26	21
Tiles Shortest Path	64.9	6	14	10	12	11	9	9	12	12	27	11	7	7
Missing Tiles	95.2	21	37	25	23	24	33	18	26	29	25	28	31	23
Tiles Composition	50.0	23	26	25	21	31	37	22	29	28	35	27	21	14

(b) Human and open-source LVLMs.