# Scaling Laws for Compositionality

**Headline graph:** Success rate as y axis, N^N attributes as x-axis,  model sizes in different colors

**Questions:**

- Is there a regular relationship between the amount of data, model size and success at compositionality?
- Does this relationship hold between modalities?
- Does this relationship differ for different attributes (number, color, shape, type etc)?
- Is there a difference between generative (e.g. synthesizing an image) and discriminative models (e.g. assessing whether a description has been met)? (This is important b.c if discriminative is easier, may allow for recursive self-play)

**Experiments:**

- **Text to Text:**
  **Dataset**
  - Synthetically create 'x is to the left of y, z is above x', 'is x to the left of z?'
  - Scale difficulty via
    - Number of objects
    - Number of dimensions (2D > 3D)

  **Model**
  - Fine tune FLAN
    - If T5, 80M, 250M, 780M, 3B, 11B
    - Could also throw in text-davinci-002
- **Text to Image (synthetic):**
  **Dataset**
  - Synthetically create object arrangements
  - Scale difficulty via
    - Number of objects
    - Color of objects
    - Size of objects (small, med, large etc)
    - Type of objects (square, triangle, circle)

  **Model**
  - Fine tune stable diffusion
  - Also try captioning model?
- **Text to Image (realistic):**
  **Dataset**
  - As above, but create from stable diffusion in-painting

  **Model**
  - As above

- **image to Text (discriminative):**
  **Dataset**
    - From above
  **Model**
    - Explore - how confident are pre-trained clip models when we sample a couple of captions from the same distribution?
    - Can we fine tune CLIP to be better at discrimination? If so, can we play a recursive improvement game?

**Operational details**

- I can source reasonable amounts of TPU time (e.g. request typical TFRC access, 5 TPUv3, 5 TPUv2) and spend ~4 hours a day programming (9pm - 1am typically)
- Looks like stable diffusion has jax implementations of stable diffusion, which IIRC are definitely the fastest way we can generate inpainting images, but also should be a great way to train. These will be worth optimizing significantly.

**Open questions**

- What is the minimum viable experiment? Is it language + fine tuning flan?
- What is the minimum viable visual experiment?
- What are Gary Marcus' recent challenges? Any we can target?

**Some considerations about text**

There are multiple ways to encode positional information. One is ‘there is x; y is to the right/left of x’ etc, describing the objects in a line LTR or RTL. The objects can also be described out of order ‘ABC-> There’s B, left of B there’s A, right of B there’s C’. This uses two concepts (is to the right of; is to the left of) whereas the first one uses one. Empirically, [redacted]’s model can tell whether two scenes described LTR or RTL are the same with 5 objects (haven’t tried more). However it gets confused when objects are described out of order. This latter can be solved if the model is first asked to write down the left to right position of the objects and then to answer the question.

Positional information can also be encoded without explicit relations; a 2x2 grid can be described left to right, top to bottom in a comma-separated way: A, B, C, D. Same for larger grids. This will be harder for models to understand but not unreasonable.

Positional info can also be given in a very random way; e.g. if one thinks of a 3x3 board, one could say that the bottom middle cell has a red circle, and that two squares up and one right from there (that is, the top right cell) has some other shape.

Tasks (need to check if these exist in bigbench or something)

- Text to text
    - Ordered scene identification: identify if two scenes described differently are the same. E.g. ABC, CBA.
        - To think about: How to express 2D (or even 3D) relationships
    - Out of order scene identification. As before but scenes can be described using arbitrary starting points; e.g. There's A, B is right of A, C is right of B. vs There's B. A is to the left of B, C is to the right of B.
    - Grid task: Given a 3x3 or bigger grid of objects (where some cells may be empty) and given a line between two points, point to the object(s) intersected
    - Question-answering: Given a scene, answer Qs. This could be multiple choice or true/false I guess
    - "Complete the pattern" (needs a better name). Say which shape/color/etc is required to e.g. form a line of green shapes, have a shape of different colors in the corners of a grid/etc
    -
- Image to text
    - Grid task, but the grid is on an image
    - [Final boss] Given a set board, say if it's valid set /jk
- Text to image
    - Given a scene description (on a grid), produce

**Gary Marcus challenges**

I was able to find these two plus the youtube video below

- **https://garymarcus.substack.com/p/form-function-and-the-giant-gulf**
- **https://garymarcus.substack.com/p/three-ideas-from-linguistics-that**
-

**Recent useful reading:**

- https://arxiv.org/pdf/2103.01403.pdf
- https://arxiv.org/pdf/2210.01603.pdf
- https://arxiv.org/pdf/1908.08351.pdf
- To Understand Language is to Understand Generalization | Eric Jang
- https://arxiv.org/pdf/2206.01714.pdf (Composable Diffusion)
- Learning to Compose Visual Relations [they point to a 'Relational CLEVR' dataset, can't find it, only the regular CLEVR]
- https://arxiv.org/pdf/1711.00350.pdf
- https://youtu.be/9tH9Qz1nH3k
- T5 paper https://arxiv.org/pdf/2210.11416.pdf
- https://arxiv.org/abs/2102.12627 (Hinton GLOM)
- (From Google Research) https://arxiv.org/pdf/2207.04901.pdf
-

To look into: PCFG (Hupkes), COGS (Kim & Linzen), SCAN (Lake & Baroni) as tests of compositionality

**Compositionality as a paradigm**

Why is this interesting? Per the papers there and as a longstanding problem in AI research, that probably inspired the early symbolic directions, compositionality is something humans leverage when dealing with various domains. Currently LLMs struggle with this because they don't learn explicit (simple algorithms) representations of concepts, the representations are encoded in the weights and can be various degrees of fuzzy; as opposed to DreamCode or similar where the concepts are stored as bag of functions that can be composed.

In humans plausibly (just off the top of my head, there's an entire research lineage that has studied this) we start with simple guesses/models that explain the data we observe, and which are held tentatively. These models are then taken as true and composed together in a domain-specific manner. E.g. we have the concept of "A is to the left of B" but this breaks if we are talking about say electrons that are very close and in superposition. These concepts are imprecise and can mean something slightly different depending on context: If we say that the apple is to the left of the bottle we probably can see both objects. In a sense San Francisco is to the left of New York (in some orientation of the globe) but this would be a rare usage of the word.

"A is to the left of B" could be taken to imply that there is nothing in between. When I say "A is to the left of B, and C is to the left of A" (CAB) I might mean that there is nothing between C and A or A and B. We typically would assume that unless told otherwise there is nothing in between. It may be worth, when specifying prompts, to try versions that fill in all the details (i.e. "if an object is said to be to the left of another, it implies there are no objects inbetween")

The number of primitive concepts implicit in a given task is not quite exactly defined. In a grid game we can say we have:

- The notions of spatial positioning
- The shape of the squares (They are not hexagons, as they are in some board games)
- That colors, shapes, etc are unique: there cannot be a blue and green object
- That the edges of the grid are such; it doesn't wrap around like a torus
- The concept of having multiple properties
- The concept that each shape exists in one square and one square only
- That squares can be either empty or have a shape
- That straight lines are supposed to be drawn from the center of the shape
- The concept of intersecting, which is less well defined for some intersections (line partially intersecting; I avoided these in my jupyter notebook after I noticed)
- …

Some of these are probably already implicit in the model but one could try making them explicit.

The grid task can be varied in a single conceptual dimension to do a scaling plot by adding more squares. It's less clear if this is done by adding a third dimension (Going from a 3x3 grid to

a 4x4 grid seems simpler conceptually than going from a 3x3 grid to a 3x3x3 grid, we're adding a spatial element. Holding the sizes constant, scaling across the 'dimensionality' axis gets us then to 3x3x3x3 grids (a tesseract task) which could be done but also requires us to think in 4D. I would recommend doing 1D lines and 2D grids for now.

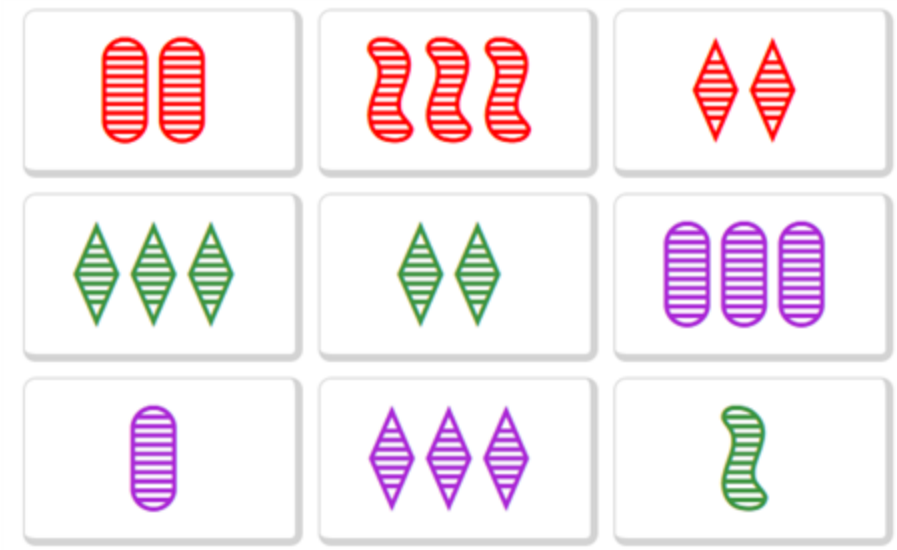
**Training/test split**
One way to finetune or train a model for these tasks is to generate a bunch of examples of the desired problem and then testing in unseen examples. This tests whether the model 'understands' the task but not necessarily if it gets the concepts right; Solving 3x3 grids should involve some skill in solving 4x4 grids too.

So the **test** task could involve solving N-order grids (same as training), but also N+1 and N+2 and seeing how those 3 values scale with model size.

**Model size**
Something I wonder is how scaling for a new task works in the context of a model that has already been trained on many other tasks vs scaling for a model that has been explicitly trained for the task at hand (i.e. a small transformer trained in a supervised way on 3x3 grids that is then tested on 4x4 grids; or performance on 3x3 grids vs supervised transformer size). What do these exponents look like in a sequence of T5-FLANs of different sizes vs smaller models? I guess we cna do the FLAN one here (because TPUs) and check on the tiny models as a stretch goal. This latter makes more sense if one were to do instead an interpretability paper to see how is it that the model is reasoning about the task rather than seeing what performance is.

**Scaling interpretation of the encoding vs scaling compositionality**
Some representations are clear in a formal sense at the cost of being less clear in a natural language sense. Comma-separated values or 2D ASCII images of grids are easier for humans to read visually but they are not natural language so models might struggle at first.

In contrast, using natural language to specify positions doesn't require the model to understand how to parse a new representation, but it could be in practice more confusing because it can be more verbose. For example, take this grid



I could say [definition of 3x3 grid] two red shaded pills, three red shaded squiggles, two red shaded diamonds, three green shaded diamonds….

Or I could say: There is A. To the right of A there is B. To the right of B there is C. Under A there is D. To the right of D there is E…

Ideally one tries both and then one can say it doesn't matter.

From ViT paper
*Position embeddings are added to the patch embeddings to retain positional information. We use standard learnable 1D position embeddings, since we have not observed significant performance gains from using more advanced 2D-aware position embeddings (Appendix D.4). The resulting sequence of embedding vectors serves as input to the encoder.*

**Paper v1 idea (Text only?)**

**Blueshift did paper**

**SD - 2d shapes for now - grammar of positions and objects**

1. **Generate one->mask->generate the rest**
2. **Generate one and then crop and insert**
3. **-> Compare qualitatitatively SD vs finetuned SD**

Goal: To measure scaling laws for compositionality across a number of tasks.

If this is the goal, then we can do this with text or images. Text is easier to evaluate success or not. But some text also has already been evaluated  (some bigbench tasks are compositionality tasks).
-> We could take those and replicate them at various T5 scales
-> If the replication goes as expected, I expect we will not find the same exponent for all the tasks (that's what the bigbench plots show). Examples:

Variable assignment & parity (Blueshift paper)

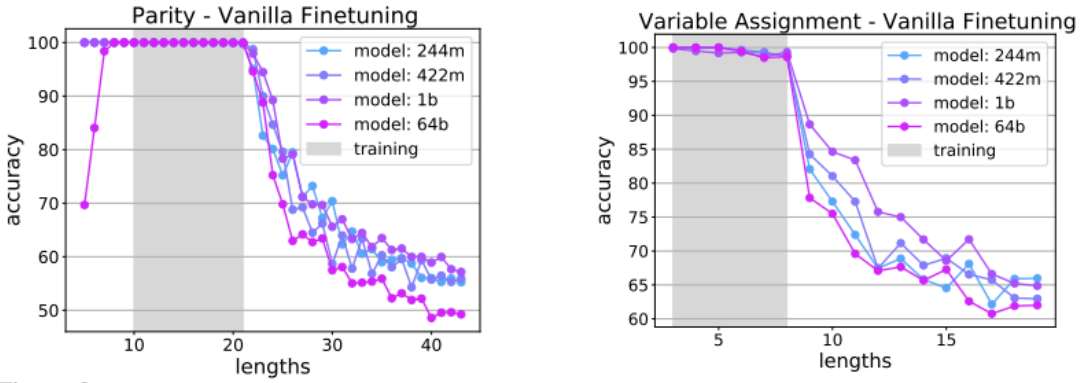



Figure 3: **Finetuned Length Generalization performance doesn't improve with scale:** Models of vastly different scales fail at length generalization on both Parity and Variable Assignment tasks, displaying identical generalization pathologies. The x-axis represents problem length and the y-axis represents the accuracy attained at that problem length. The training lengths are highlighted in grey.