Compositional learning of functions in humans and machines

Yanli Zhou[1,3]
yanlizhou@nyu.edu
[1]Center for Data Science
New York University

Brenden M. Lake[1,2]
brenden@nyu.edu
[2]Department of Psychology
New York University

Adina Williams[3]
adinawilliams@meta.com
[3]FAIR Laboratories
Meta AI

Abstract

The human ability to learn and compose conceptual operations is foundational to making flexible generalizations, such as creating new dishes from known cooking processes. Beyond naive chaining of functions, there is evidence from the linguistic literature that people can learn and apply context-sensitive, interactive rules, such that output production depends on context changes induced by different function orderings. Extending the investigation into the visual domain, we developed a function learning paradigm to explore the capacity of humans and neural network models in learning and reasoning with compositional functions under varied interaction conditions. Following brief training on individual functions, human participants were assessed on composing two learned functions, in ways covering four main interaction types, including instances in which the application of the first function creates or removes the context for applying the second function. Our findings indicate that humans can make zero-shot generalizations on novel visual function compositions across interaction conditions, demonstrating sensitivity to contextual changes. A comparison with a neural network model on the same task reveals that, through the meta-learning for compositionality (MLC) approach, a standard sequence-to-sequence Transformer can approximate a strong function learner, and also mimic human error patterns with additional fine-tuning.

Keywords: function composition; meta-learning; compositional generalization; order of operations

Introduction

Humans are efficient and flexible learners, with the ability to infer an underlying functional operation from exposure to just a few input-output examples. Humans are also adept in flexibly combining previously learned functions in new ways. For example, someone who knows how to chop vegetables and knows how to fry food can learn how to make fries, by chaining their chopping skills with their frying skills, even if they have never previously attempted that dish. Moreover, function composition skills emerge early in life, with children beginning to learn how to compose visual functions without explicit training at as early as 3.5 years of age (Piantadosi & Aslin, 2016). Mastery of these skills over time becomes a cornerstone for comprehending complex symbolic systems, fostering abstract thinking and aiding the acquisition of conceptual knowledge (Curry & Feys, 1958; Schönfinkel, 1967; Piantadosi et al., 2012).

Beyond the sequential application of functions, humans can also track contextual changes that occur during function composition as a result of the order of operations. In the previous example, if the potatoes have been puréed, one would quickly realize that the frying operation can no longer take place, as the puréeing function has transformed the potatoes into a state unsuitable for the second function to apply. Linguists have formalized different context-shifting phenomena into four types of function ordering (Kiparsky, 1968): *feeding* describes scenarios in which the application of the first function creates the context for the second one to apply, as illustrated in the example when *chopping* feeds the function *frying* (Fig.1, top); *counter-feeding* describes the reverse order of application, with the context-creating function applied too late. For example, trying—and failing—to fry whole potatoes before chopping them. *Bleeding* occurs when the context of the second function is removed by the first function's application, akin to the example where *puréeing* potatoes makes the unsuitable for later *frying* (Fig.1, bottom); counter-bleeding is its reverse, where the context-removal happens after the first function has successfully been applied (i.e., *frying* does not prevent one from later *puréeing*). Together, we see that function composition emerges as a challenging learning program, requiring not only generalization to novel instances on the individual function level, but also to compositions of two functions with sensitivity to different interactions induced by various function orders.

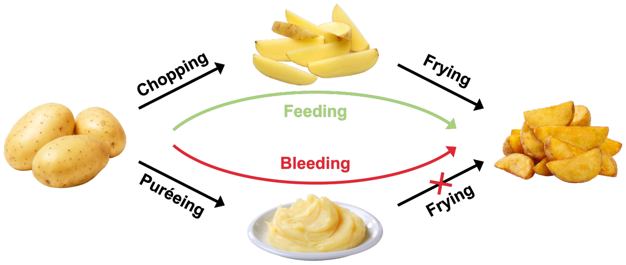

Figure 1: Function interactions influence function compositions. The chopping operation provides context for frying while the puréeing operation renders frying inapplicable.

Past research has investigated whether models can capture human-like compositional learning and generalization. For instance, Lake and Baroni (2018) proposed the SCAN task to assess the compositional skills of neural network models. The authors, as well as in many subsequent investigations (Bastings et al., 2018; Ruis et al., 2020; Valvoda et al., 2022), found that neural networks continue to struggle with systematic generalizations despite recent AI advances. Liška et al. (2018) introduced a more explicit test of function composition, and found that only a small portion of tested recurrent neural networks converged to compositional solutions in computing the outputs of composite lookup tables. Previous studies have also evaluated model performance alongside empirical investigations, including explorations of how humans learn structured visual concepts by reasoning about how parts compose (Overlan et al., 2017; Zhou et al., 2024). More recently, Lake and Baroni (2023) showed that humans excel at composing functions in instruction learning, and demonstrated that a standard neural network-based model can be optimized to exhibit human-like behavior through a meta-learning procedure that encourages compositionality. Despite varying levels of modeling success reported in previous work on compositional function learning, none of them systematically studied the learning of functions and their interactions in humans and machines.

Towards this end, we proposed a learning paradigm suitable for testing both humans and models on their compositional function learning skills, with a special focus on the variety of interactions in sequential function transformations. Extending the experimental framework from Piantadosi and Aslin (2016), we investigated whether participants could learn visual functions as transformations of cartoon cars moving in and out of factory units with minimal input-output exposure. For each prompted input car, participants were evaluated on zero-shot function composition in feeding, counter-feeding, bleeding, and counter-bleeding conditions. Our experiment demonstrated that humans efficiently generalize from newly learned single functions to their compositions, achieving consistently high levels of accuracy across different orderings of visual functions, contrary to previous linguistic theories on human learning biases in function interactions. Our results suggested that humans are sensitive to contextual changes during function composition, generating different outputs based on the order of operations.

Following Lake and Baroni (2023), we trained a neural network through meta-learning for compositionality (MLC) in order to learn functions and their compositional interactions. When comparing model performance directly to behavioral data on the same learning task, we found that a standard Transformer (Vaswani et al., 2017), without any explicit tooling for symbolic reasoning, can be trained to simulate a strong, compositional function learner through a series of function composition tasks. Additionally, after fine-tuning on human and curated data exhibiting particular biases (McCoy & Griffiths, 2023; Zhou et al., 2024), the model produced human-like error patterns such as reversal of function orders, showing that additional processes operating on top of idealized function learning best explain the human behavior.

Behavioral experiment

We retrofitted the task design introduced by Piantadosi and Aslin (2016) with an extended space of functions and features suitable for testing both humans and models in various interaction conditions. Specifically, participants took part in an online experiment titled "car assembly game" in which they played the role of production line workers in a car factory. The task was to assemble cars based on the knowledge of the different factory units installed on each assembly line.

Stimuli generation. The main class of stimuli consisted of programmatically generated cartoon cars (Fig.2A). Each car is represented internally as a tree with its root node being the car body. The car body can have up to 3 different child nodes denoting the window, tires or lights respectively. Each car part also has two child nodes representing its type and color. There are 3 possible types for each car part, reflecting different part shapes, and 7 possible, colorblind safe colors, including a "no color" option. This defines a space of 375,000 distinct car objects.

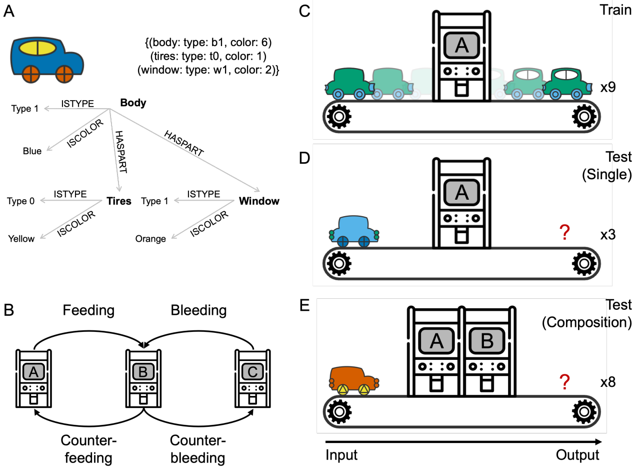

Figure 2: Stimuli and experimental procedure. (A) Example of a procedurally generated car stimulus. (B) Participants learn 3 functions *A*, *B* and *C* satisfying a set of 4 interaction relations. Arrows denote the order of function application (see concrete examples in Fig.4). (C) During the training period, participants saw 9 cars moving through each factory unit representing a different function. If a car is a valid input to the function, it will come out of the unit with changes reflecting the underlying function. Cars remain unchanged if they are invalid inputs. (D) Participants were first asked to generate the correct output based on each prompted input car and factory unit. (E) For each of the interaction types, represented by the relevant factory units and the order of their application, participants were asked to generate the correct output car for 8 different input cars.

All functions are defined as tree operations on the car objects. For example, a function describing the action "adding an oval shaped window to a car without any existing window" can be written as

```
if not HAS(car.body, window) then
    ADDCHILD(car.body, CarNode(window,
     style=1, color=None)).
```

Other possible edits to the car are *remove* by deleting a specific child node of the car body, *paint* by changing the color of a child node, and *editpart* by changing the type of a child node. We limit the scope of the functions considered in this experiment to target only one child node of the car body at a time. The condition of all function applications are limited to targeting the same part of the car in which the function transformation takes place. The resulting functional space consisted of 1,081 distinct functions.

Task procedure. The experiment was split up into a training stage and a testing stage. During the training stage, participants were familiarized with 3 individual functions each represented by a different factory unit (*A*, *B* or *C*). The functions were chosen to satisfy a set of pairwise interaction relations (Fig.2B) based on the order of application. The pair *AB* always represents a feeding relationship, with its reverse *BA*

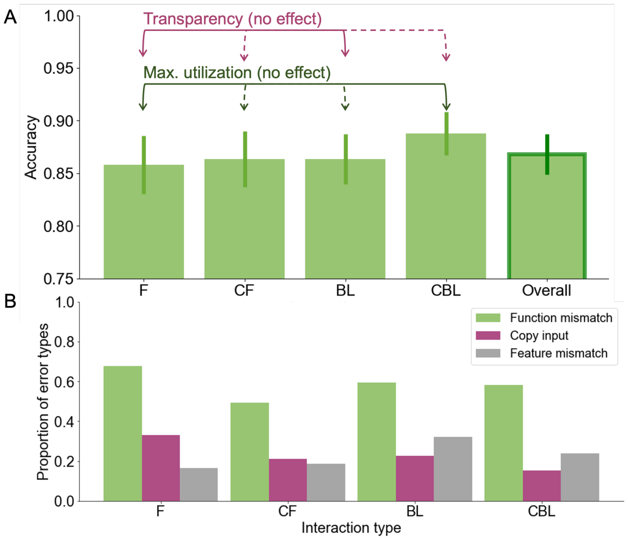

Figure 3: Behavioral results. (A) Mean generation accuracy and standard error by interaction type. Performance did not differ significantly across different conditions. We also did not observe higher performance in F/CBL trials over CF/BL trials (maximum utilization bias), nor F/BL over CF/CBL trials (transparency bias). (B) Proportion of each error type over all incorrect generations by interaction type.

representing a counter-feeding relationship; the pair *CB* always satisfies a bleeding relation, with *BC* representing its reverse (see example trials in Fig.4). Eight possible function triplets were pre-generated, and each participant was randomly assigned one of the sets.

Participants observed 9 unique cars moving through each factory unit on a conveyor belt (Fig.2C). If a given car was a valid input to the function, it emerged from the unit with one part transformed; otherwise, the car remained unchanged. After the presentation of exemplars for each factory unit, a short test followed to determine whether participants inferred the expected underlying single function. Specifically, we asked them to assemble the correct output car for each of 3 prompted input cars using the game interface (Fig.2D). All example input-output pairs remained on screen throughout this period, and participants were informed of the correctness of their generations. For each input car, participants were given up to 3 opportunities to configure the correct output.

Finally, participants were asked to generate the correct output for each prompted input car based on two different factory units shown in consecutive orders (Fig.2E). Participants received no training on any of the function pairs. No feedback was provided during this phase. For each of the interaction pairs *AB*, *BA*, *CB* and *BC*, participants completed a block of 8 output generations sequentially. The order of each block was randomized for each participant, and the prompted input cars were also presented in a randomized order. Throughout the testing stage, participants had access to all previously seen examples of each relevant function.

Participants. We recruited participants via Amazon Mechanical Turk for the online experiment. To recruit high quality participants, we first conducted a pre-experiment survey consisting of 10 simple attention-checking questions about the car assembly game setup. Of the 196 workers who responded to the survey, we invited those who scored over 90% ($n = 117$) to participate in the main experiment. Participants completed the main task within an hour and 30 minutes, and were compensated $14.00, plus up to $2.00 of performance-based bonus ($2.00 × overall accuracy).

Behavioral results

The first analysis examined how people performed on few-shot learning of individual, isolated functions. Specifically, during the testing stage, participants generated the expected responses for most of the prompted inputs to individual (non-compositional) functions ($M = 95.4\%$, $SEM = 0.010$). The three input cars for each single function were analyzed based on three groups: *familiar*, *novel*, and *identity*. Cars in the *familiar* group consisted of example inputs seen during the training stage, which checked for attention lapses and familiarized participants with the generation interface. The *novel* group contained novel cars that are valid inputs to the considered function, and *identity* items were invalid inputs and identical copies of the input should be returned. The mean accuracy was 95.2% ($SEM = 0.012$), 95.7% ($SEM = 0.012$) and 95.4% ($SEM = 0.013$) for these groups respectively. Together, high accuracy on all three groups suggests sufficient learning of the single functions.

The second analysis examined how people reasoned about function composition under interaction. During test, there were four consecutive blocks, each representing one of the feeding (F), counter-feeding (CF), bleeding (BL), and counter-bleeding (CBL) interaction conditions (Fig. 3A). Participants achieved an average accuracy of 86.8% ($SEM = 0.019$) across all trials and conditions. Additionally, we found that participants performed comparably well across interaction conditions, each with a mean generation accuracy between 85.8% and 88.8%. Using test conditions as fixed effects while accounting for the random effect of participant ID, a mixed effects logistic model on generation correctness was performed. The analysis did not reveal a significant effect for each of the interaction conditions ($\beta_{BL} = -1.609, SEM = 1.111, p = 0.148$; $\beta_{CBL} = -1.904, SEM = 1.063, p = 0.073$; $\beta_{CF} = 1.109, SEM = 1.429, p = 0.438$).

We also evaluated whether biases suggested in past linguistic studies are observed in the current study in visual function learning. Kiparsky (1968) proposed the *maximum utilization* bias, whereby feeding and counter-bleeding are easier to learn because both functions apply. Kiparsky (1971) also later discussed a *transparency* bias that favors instead the more transparent interactions of feeding and bleeding, over the more opaque interactions counter-feeding and counter-bleeding. As we did not observe significant difference across interaction conditions, the results also did not suggest an active maximum utilization bias ($t(116) = 0.488$, $p = 0.627$) nor a transparency bias ($t(116) = -0.951$, $p = 0.343$).

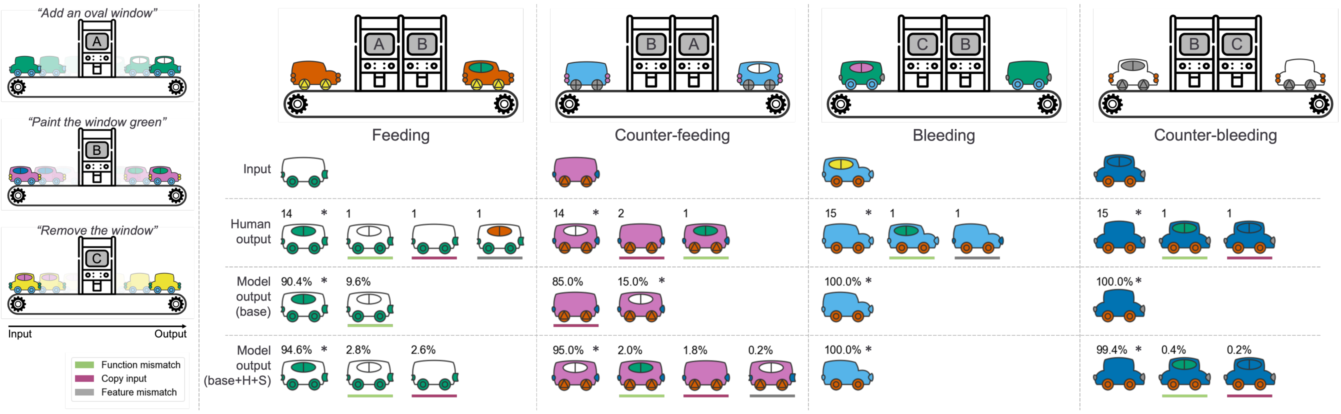

Figure 4: Examples of human and model output for each interaction type. Individual functions *A*, *B* and *C* are shown on the left. Prompted input cars (top rows) are shown alongside different output generations by humans, the base MLC model, and the fine-tuned MLC. The number in the top-left corner reflects the count (or percentage for model samples) of generation for each car. Correct outputs are marked by $*$; erroneous generations are underlined with colors corresponding to the error types (green: *function mismatch*; purple: *input copying*; grey: *feature mismatch*).

Error analysis For each composition of two functions, we computed the key transformation expected from input to output. For instance, composing both functions *A* and *B* results in the addition of a green oval-shaped window (Fig. 4). The majority of human-generated output cars were correct, and reflected the key transformation expected in all interaction conditions. The majority of incorrect generations fall into three main error types observed in the behavioral data (Fig.3B). The most frequent type of error, which we term *function mismatch*, describes the scenario in which a participant fails to reflect the key transformation in their generation, by either applying only one of the functions to the input or by reversing the order of function application. For instance, participants sometimes only applied the first function to the input, and ignored the second function in the feeding condition, leaving the newly attached window unpainted (left-most column, green underlines in Fig. 4). There were also cases where participants only applied the second function, as seen in the bleeding examples in Fig. 4 of cars that had a green window rather than no window. In the counter-feeding example of Fig. 4, some participants erroneously applied function *A* before *B*, resulting in a green window in their generations. Another type of error, *input copying*, occurs when the participant simply makes an identical copy of the input (see examples in Fig. 4, purple underlines)[1]. Directly copying the input is a strategy some participants use to produce an output visually similar to the expected answer without applying any function transformation. Finally, for *feature mismatch* errors, participants correctly reflected the key transformation but made a mistake in a part of the car irrelevant to the functions in Fig. 4, grey underlines). We see that despite high performance overall, participants sometimes deviate from perfect compositional behavior in these 3 principal ways.

Composing functions with meta-learning

To model human behavior in the function composition task, we trained a neural network for learning compositional functions following the meta-learning for compositionality (MLC) approach (Lake & Baroni, 2023). Similar to the motivation in McCoy and Griffiths (2023), we use meta-learning to study human inductive biases at Marr's computational level (Marr, 1982). To do so, we simulate idealized function learning by forming a generative process over systems of functions, and using samples from this process as meta-learning episodes for the neural network. To examine ways in which people deviate from idealized function learning, we further enriched the network with additional biases during the meta-learning process (Lake & Baroni, 2023; Zhou et al., 2024).

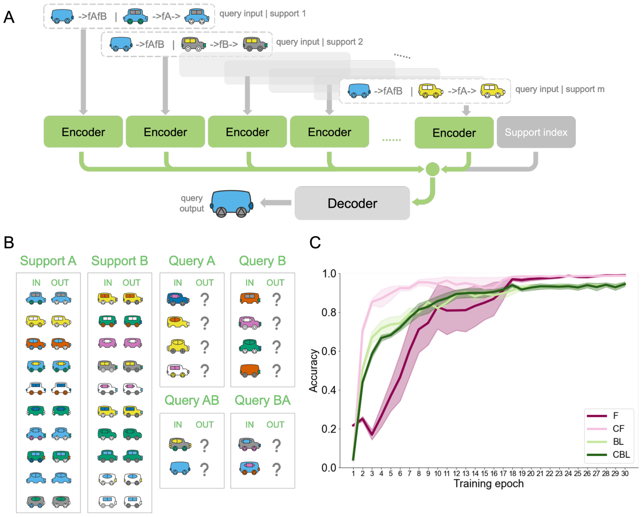

Figure 5: (A) Model schematics. (B) Example of a typical episode; After observing a set of support input-function-output tuples, the model is queried on both single functions and their compositions. (C) Typical learning curves for different interaction types over the course of base training.

[1] In the current task setup, input-copying is sometimes indistinguishable from *function mismatch* errors. For example, in the counter-feeding examples in Fig. 4 that are purple-underlined, it is unclear whether participants intended to explicitly copy the input as a strategy, or whether they tried applying only the first function, ending up with an identical copy due to input invalidity.

Model description. A schematic of the MLC model structure is in Fig. 5A. The MLC model uses a standard sequence-to-sequence Transformer trained from scratch and optimized to generate the correct output in response to a novel query based a set of support examples (Lake & Baroni, 2023). To process the study examples and the queries, an encoder network takes as input a sequence of strings each representing the input car, the function handle and the corresponding output car for each support example (e.g., $\text{car}_{in} \rightarrow fA \rightarrow \text{car}_{out}$), and only the input car and function handle(s) for each query (e.g., $\text{car}_{in} \rightarrow fAfB \rightarrow$). The tree structure of each car object is flattened into a string representation to form each car_{in} and car_{out} (see an example in Fig.2A), before concatenation with function handles indicating the participating functions and the "→" separator in between. Each query is duplicated and paired with each of the m study examples per episode forming m input sequences, which are processed by the shared encoder into m separate latent embeddings. The set of latent embeddings are then each combined with an index embedding indicating its original place in the support set, before being processed together by a decoder network that generates the output sequence car_{out}. The encoder and the decoder are both 2-layer Transformers with 8 attention heads in each layer; additional training hyper-parameters are set to default values used in Lake and Baroni (2023).

Base training. Guiding the model to strong function learning capabilities, we train the MLC model via meta-learning (Hospedales, Antoniou, Micaelli, & Storkey, 2022), encouraging the model to develop task-general compositional skills through a series of episodes that each constitutes a different function composition task. The objective of each task is to generate the correct output of a novel composition of two functions, given the input and a limited set of support input-output examples for each function. Each episode consisted of (1) support input-output pairs of two functions A and B; (2) A set of query inputs for the single functions A and B, as well as the compositions AB and BA (see Fig.5B). This setup is comparable to the behavioral experiment, where participants were trained on two functions and tested on their compositions.

To form the episodes used in base training, we first divided all possible 1,081 car-edit functions into a training set and a held-out validation set (with no individual functions appearing in both). We also withheld all functions used in the behavioral experiment from the training. To generate each training episode, we randomly sampled two functions (arbitrarily labeled A and B) from the training distribution that either satisfy a feeding/counter-feeding relationship or a bleeding/counter-bleeding relationship. For each function, we sampled 6 valid support input-output examples, and 4 invalid examples (input and output are identical). To train the model on generalizing to both new input of learned single functions, as well as novel function compositions and interactions, we queried the model with (1) novel inputs for each single function (e.g., $\text{car}_{in} \rightarrow fA \rightarrow$), of which 2 are valid and 2 are invalid inputs; (2) inputs to function compositions (e.g., $\text{car}_{in} \rightarrow fAfB \rightarrow$), two for $fAfB$, and two for $fBfA$. All query items were randomly sampled from all possible inputs to the relevant function(s) given validity constraint. We generated 50,000 training episodes in total, half of which contained feeding/counter-feeding function pairs and the other contained bleeding/counter-bleeding functions. With a batch size of 5 episodes, training optimized the cross-entropy loss (averaged over tokens). Over the course of training (50 epochs), the model must learn to represent single functions based on the support examples, and to compose two functions according to the order of function application, all without explicit compositional supervision.

	Log-likelihood	Acc. (Overall)	Acc. (F)	Acc. (CF)	Acc. (BL)	Acc. (CBL)
Human		85.7% (±0.029)	**84.7%** (±0.039)	85.5% (±0.037)	85.1% (±0.034)	87.5% (±0.034)
MLC (Base)	-2.817	81.9% (±0.022)	81.5% (±0.042)	80.1% (±0.063)	77.9% (±0.029)	88.1% (±0.050)
MLC (Base+H)	-1.830	83.0% (±0.047)	73.2% (±0.056)	87.7% (±0.040)	80.8% (±0.050)	90.1% (±0.038)
MLC (Base+H+S)	**-1.596**	**88.6%** (±0.062)	82.3% (±0.048)	**88.7%** (±0.041)	**88.9%** (±0.040)	**94.5%** (±0.023)

Figure 6: Modeling results. (Column 1) Model goodness of fit for predicting human generated examples. For each model version, the overall average log-likelihood per human generated output car is in the first column. (Column 2-6) Human vs. model accuracy by interaction type on held-out experiment trials. Bolding indicates best performance.

Fine-tuning with additional biases and human data. Given that people were not perfect function learners in the behavioral experiment, to study these deviations we fine-tuned the MLC model on two additional distributions. The first data distribution H was formed using raw human data. We used human responses on 4 out of the 8 total function triplets assigned to participants in the behavioral experiment for additional training, and reserved the rest for evaluation. Each episode in H was structured identically to the base training episodes, except that support input-output pairs and queried items were replaced with the ones used in the behavioral experiment. Since participants were trained on 3 single functions, each participant's data was used to form two separate episodes, one for feeding/counter-feeding trials, and the other for bleeding/counter-bleeding trials. As a result, the H distribution contained 110 episodes, each with 18 support examples (9 for each function), 6 single function queries (3 for each function) and 16 composition queries (8 for each function composition). Additionally, to further examine the sub-optimal heuristics at work, we created a synthetic data distribution S containing 2,000 episodes with simulated function mismatch behavior. Specifically, to capture function misapplication (as in Fig. 4 green and purple underlines), we modified the generative process such that output sequences to the queries was generated with a reversed function order p_{flip} of the times. Together, the model parameters were fine-tuned to predict the outputs in H and S, instead of the ground-truth.

Modeling results Using the reserved validation function set, we created a set of 3,000 base validation episodes. At the conclusion of base training, the MLC model generated the correct output sequence for each query at an average of 97.9% accuracy across query types and different random initializations ($SEM = 0.037$). When queried on single functions, the model was able to make near-perfect generalizations ($M = 98.1\%$, $SEM = 0.004$). When divided into interaction types, validation accuracy on function compositions remained consistently high ($M_F = 97.6\%$, $SEM_F = 0.007$; $M_{CF} = 98.0\%$, $SEM_{CF} = 0.004$; $M_{BL} = 97.3\%$, $SEM_{BL} = 0.004$; $M_{CBL} = 97.1\%$, $SEM_{CBL} = 0.005$).

To directly compare the generalization behavior of the model with humans, we evaluated generations from MLC models on the set of experimental trials using the 4 held-out function triplets. Model accuracy results are summarized in Fig. 6. We observed slightly lower overall accuracy from the MLC model with only base training. However, our primary modeling goal here is not to train a model that surpasses humans on these trials but to provide a more detailed account of human compositional generalization behavior. Reviewing the examples of model samples shown in Fig. 4, we observe that the base MLC model can produce the correct output most of the time, but sometimes also makes errors such as copying the input, echoing the strategy some participants used to generate visually similar output items without applying any function transformation. However, base MLC samples do not reflect the full variety of human-like errors, rarely displaying function or feature mismatch behavior. After fine-tuning with only human-generated data (H), a minimal number of training episodes compared to the base training set, the MLC model demonstrated an improved account of held-out data in terms of log-likelihood. When MLC model was provided with additional guidance from both H and S, with S introducing a target bias previously not encoded in base training, it performed the best on held-out data and exhibited an overall improvement in log-likelihood over the base MLC model (Fig. 6). Qualitatively, the fine-tuned MLC model shows a closer match to the human behavioral outputs (Fig. 4), showing that error-prone function application, added to the training of a strong neural network learner, provided the best account of the human behavior.

Discussion

To study zero-shot visual function composition in both humans and machines, we designed a learning paradigm to examine generation behavior under different interaction conditions. Specifically, through training participants on small sets of input-output examples, we observed highly accurate generalizations to novel function inputs. During the test phase, we presented participants with queries involving the four logical patterns that result from combining pairs of interacting functions. We report a consistently high level of generation accuracy across all conditions, suggesting that humans can adeptly handle contextual shifts when composing functions. Follow-up investigations with more challenging functions might provide a better picture of the learnability of different interaction types. While we do not observe overall mean differences across conditions, we observed that humans make non-random, structural mistakes when processing two functions consecutively: partial application of functions or a reversal of function order constitute the main families of errors.

In a side-by-side evaluation of human and machine performance, we found that a standard sequence-to-sequence Transformer can be trained to compose novel function pairs via meta-learning on streams of function composition tasks. In particular, our MLC model was capable of learning both single functions from limited examples and generalizing to unseen function composites by producing output sequences to promoted inputs at a near-human accuracy. We used error patterns uncovered in behavioral data analysis to form the basis of comparison with model error patterns, and further revealed how distant models are from humans in understanding function interactions. To further test for the different heuristics driving behavior divergent from an idealized function learner, we injected a function mismatch bias into a synthetic training distribution, along with additional examples of human behavior. After fine-tuning, the model generations demonstrated a fuller variety of human-like error patterns. Improved behavioral fit with fine-tuning suggest that human behavior is most consistent with error-prone function application on top of an otherwise strong, approximate function learner.

Although we found relatively uniform levels of performance across conditions in both behavioral and computational analyses, the various learning trajectories of the 4 function interaction types during base model training might shed light on how different processes are mastered over the course of development (Fig.5C). While validation performance ultimately converged to high levels at the end of training, feeding was clearly the hardest for the model to master, followed by bleeding and counter-bleeding, with counter-feeding learned most early and easily. This result was expected as feeding always requires the model to apply two function transformations when the input is valid, while in other cases only one function application takes place at times. However, it is interesting to note that counter-feeding, or the application of only the second function in a novel composite, is also demonstrated to be an infant behavior before the onset of function composition skills (Piantadosi, Palmeri, & Aslin, 2018). Extending the current study to evaluate populations at various developmental stages might help elucidate if certain function interaction types are indeed harder to grasp.

As a future step, sequences currently used by our model can be readily translated into other representational formats such as raw images, which are what human participants observed, or tree structures, which do not enforce arbitrary ordering of car parts and features. More direct comparisons between human and model behavior can offer additional insight on how function interactions are learned and processed, and may further inform how to build computational systems with more human-like compositional learning.

Acknowledgements

This work was supported by NSF Award 1922658 NRT-HDR: FUTURE Foundations, Translation, and Responsibility for Data Science. Yanli Zhou was supported by the Meta AI Mentorship Program. We thank Solim LeGris and Cindy Luo for helpful discussions of this manuscript.

References

Bastings, J., Baroni, M., Weston, J., Cho, K., & Kiela, D. (2018). Jump to better conclusions: SCAN both left and right. *EMNLP 2018 - 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, Proceedings of the 1st Workshop*, 47–55.

Curry, H. B., & Feys, R. (1958). *Combinatory logic* (Vol. 1). North-Holland Amsterdam.

Hospedales, T., Antoniou, A., Micaelli, P., & Storkey, A. (2022). Meta Learning in Neural Networks: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Kiparsky, P. (1968). Linguistic universals and linguistic change. In E. Bach & R. Harms (Eds.), *Universals in linguistic theory* (pp. 170–202). Holt, Rinehart, and Winston.

Kiparsky, P. (1971). Historical linguistics. In W. O. Dingwall (Ed.), *A survey of linguistic science* (pp. 576–642). College Park: University of Maryland Linguistics Program.

Lake, B. M., & Baroni, M. (2018). Generalization without Systematicity: On the Compositional Skills of Sequence-to-Sequence Recurrent Networks. In *International Conference on Machine Learning (ICML)* (pp. 2873–2882).

Lake, B. M., & Baroni, M. (2023). Human-like systematic generalization through a meta-learning neural network. *Nature*, *623*(7985), 115–121. doi: 10.1038/s41586-023-06668-3

Liška, A., Kruszewski, G., & Baroni, M. (2018). *Memorize or generalize? searching for a compositional rnn in a haystack.*

Marr, D. C. (1982). *Vision*. San Francisco, CA: W.H. Freeman and Company.

McCoy, R. T., & Griffiths, T. L. (2023). *Modeling rapid language learning by distilling bayesian priors into artificial neural networks.*

Overlan, M. C., Jacobs, R. A., & Piantadosi, S. T. (2017). Learning abstract visual concepts via probabilistic program induction in a Language of Thought. *Cognition*, *168*, 320–334.

Piantadosi, S. T., & Aslin, R. (2016). Compositional reasoning in early childhood. *PLoS ONE*, *11*, 1–12.

Piantadosi, S. T., Palmeri, H., & Aslin, R. (2018). Limits on Composition of Conceptual Operations in 9-Month-Olds. *Infancy*, *23*, 310–324.

Piantadosi, S. T., Tenenbaum, J. B., & Goodman, N. D. (2012). Bootstrapping in a language of thought: A formal model of numerical concept learning. *Cognition*, *123*, 199–217.

Ruis, L., Andreas, J., Baroni, M., Bouchacourt, D., & Lake, B. M. (2020). A Benchmark for Systematic Generalization in Grounded Language Understanding. In *Advances in Neural Information Processing Systems 33.*

Schönfinkel, M. (1967). On the building blocks of mathematical logic. *From Frege to Gödel*, 355–366.

Valvoda, J., Saphra, N., Rawski, J., Williams, A., & Cotterell, R. (2022, October). Benchmarking compositionality with formal languages. In N. Calzolari et al. (Eds.), *Proceedings of the 29th international conference on computational linguistics* (pp. 6007–6018). Gyeongju, Republic of Korea: International Committee on Computational Linguistics. Retrieved from https://aclanthology.org/2022.coling-1.525

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., … Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

Zhou, Y., Feinman, R., & Lake, B. M. (2024). Compositional diversity in visual concept learning. *Cognition*, *244*, 105711. doi: https://doi.org/10.1016/j.cognition.2023.105711