Out-of-Distribution Generalization in Algorithmic Reasoning Through Curriculum Learning

Andrew J. Nam
Department of Psychology
Stanford University
ajhnam@stanford.edu

Mustafa Abdool
Department of Computer Science
Stanford University
moose878@gmail.com

Trevor Maxfield
Institute of Computational and Mathematical Engineering
Stanford University
maxfit@stanford.edu

James L. McClelland
Department of Psychology
Stanford University
jlmcc@stanford.edu

Abstract

Out-of-distribution generalization (OODG) is a longstanding challenge for neural networks, and is quite apparent in tasks with well-defined variables and rules, where explicit use of the rules can solve problems independently of the particular values of the variables. Large transformer-based language models have pushed the boundaries on how well neural networks can generalize to novel inputs, but their complexity obfuscates how they achieve such robustness. As a step toward understanding how transformer-based systems generalize, we explore the question of OODG in smaller scale transformers. Using a reasoning task based on the puzzle Sudoku, we show that OODG can occur on complex problems if the training set includes examples sampled from the whole distribution of simpler component tasks.

Large transformer-based 'foundation' models [1] have attracted recent attention by showing some success in mathematical reasoning tasks, demonstrating a degree of systematicity and compositionality [2, 3, 4]. However, it is unclear how their ability to behave systematically emerges, due to the massive sizes of the training data and model parameters. Are they demonstrating the ability to generalize out-of-distribution to novel problems? Or are they succeeding because the data they are trained on samples from the entire space of possible training examples?

To investigate how a domain-agnostic model may learn to generalize to out-of-distribution examples, we train a transformer-based network to learn a simple solution strategy to the popular puzzle game Sudoku. We use a 6x6 Sudoku grid rather than the traditional 9x9, which provides sufficient complexity for investigating algorithmic reasoning while offering more tractability and lower compute requirements. The general rule of Sudoku still applies: every n-celled row, column, and outlined region of the grid must contain exactly one instance of each of the n alternative digits.

Sudoku is well-suited for this inquiry for several reasons. First, it is governed by a small set of rules that are inherently abstract, relational, and form sophisticated interactions and dependencies that require careful algorithmic and deductive reasoning. These rules form group properties and symmetries [5, 6] that translate one puzzle to another such that learning to solve a subset of examples of a class of puzzles would enable the solver to solve all puzzles with the same relational properties, provided the abstract rules and relations are correctly induced. This enables an elegant means to probe for systematicity by designing training and test sets that share core relational features yet differ superficially in a well-defined manner. Second, Sudoku has been shown to be challenging to neural networks, and has only been successfully solved using a graph network architecture [7] with built-in

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

domain-specific inductive biases enforcing the relevant, task-specific, symmetries [8]. While this is a useful strategy for building models that solve Sudoku, it offers little towards understanding how a domain-agnostic neural network can learn the underlying rules in a way that would enable systematic generalization. Finally, Sudoku has been used to study reasoning, learning, and generalization in humans [9, 10], offering an interesting benchmark for what forms of behavior one ought to expect from a solver with human-level general intelligence.

We focus on one solution strategy in Sudoku called the Hidden Single technique and introduce a transformer neural network architecture and training set to explore out-of-distribution generalization (OODG). Building on this network, we present the following findings: First, a single forward pass in the network is insufficient to learn the Hidden Single strategy; sequential, multi-step reasoning is necessary. Second, we decompose the Hidden Single strategy into two subtasks, and show that including training examples on these subtasks sampled from the full space of instances of such problems allows the model to exhibit substantial OODG.

1 Task Description

We design base our tasks on the three simplest Sudoku techniques. *Full House (FH)* involves identifying a cell in which all other cells in one of its houses (row, column, or 2x3 box) are filled such that the empty cell's digit must be the only remaining digit. *Naked Single (NS)* involves identifying a cell in which 5 of the 6 possible digits already exist in its neighborhood (row, column, and 2x3 box) such that the empty cell's digit must be the remaining digit. *Hidden Single (HS)* involves identifying a cell C in which all other cells c_i in one of its houses cannot contain one of the digits, either due to c_i already containing a different digit or the digit being present in c_i's neighborhood, such that the only remaining cell that can contain the digit in the house is C.

For each technique, we create a task in which the model is presented with a 6x6 Sudoku grid and a string sequence prompt that provides the context for solving the problem, including the coordinates of the cell to solve for, the candidate digit, the name of the technique to use, and, for the Hidden Single (HS) and Full House (FH) tasks, the house type to inspect. By specifying all these details as part of the prompt, we simplify the task from conducting a search for a valid solution to verifying whether a goal cell should contain a candidate digit according to the rules of the specified technique.

The target sequence formats for each of the three tasks were designed to support composition of elements of the Full House and Naked Singles (NS) tasks (see Table 1). The HS task format steps through all of the cells other than the target cell in the specified target house, and checks to see if the specified digit can go in any of these cells. If it cannot, the answer is yes, it must go in the target cell.The FH task performs a similar iteration strategy, but identifies whether the cell contains a digit at all at each step. The NS task matched the sub-tasks in the HS task, requiring the model to identify whether a given cell can contain a candidate digit based on any direct contradictions within its shared row, column, or box neighborhood.

Table 1: Sample problems. Note that rows count top to bottom and columns count left to right.

	5		6	2	
2		6	1	3	5
	6		2	5	1
5			3		6
6			5	1	3
		5		6	2

	Hidden Single	Full House	Naked Single
Prompt	<SOS>hidden_single goal_cell row 6 column 2 house_type column digit 3 can_contain	<SOS>full_house goal_cell row 2 column 2 house_type box digit 4 is_filled	<SOS>digit 6 can_contain row 4 column 3
Target	row 1 column 2 no row 2 column 2 no row 3 column 2 no row 4 column 2 no row 5 column 2 no solution yes <EOS>	row 1 column 1 no row 1 column 2 yes row 1 column 3 no row 2 column 1 yes row 2 column 3 yes solution no <EOS>	no <EOS>

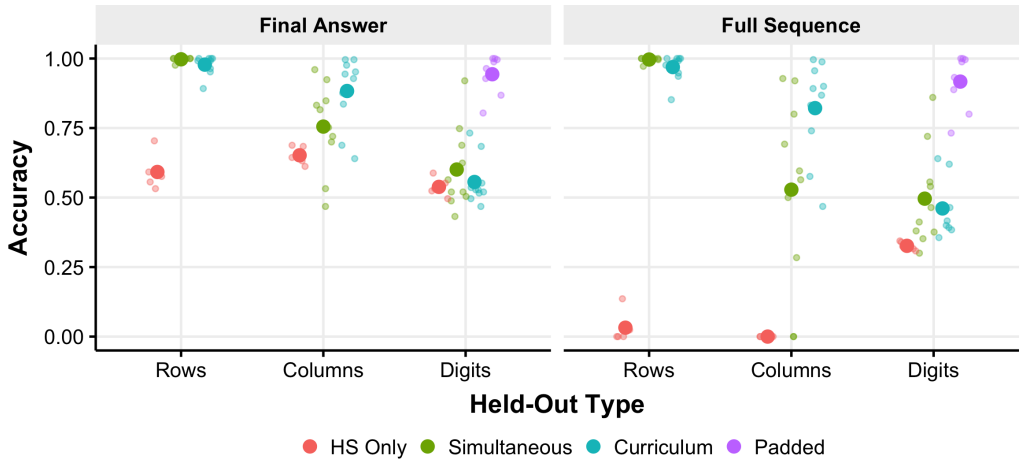

Figure 1: Out-of-distribution accuracy results. Small dots represent individual models (10 per condition). Large dots represent average accuracy in each condition. Left: accuracy based on final yes/no at the end of output sequence. Right: accuracy based on the entire output sequence.

2 Experiments

We use a 3-layer transformer encoder [11] to which all grid and text embeddings are passed, and from which output vectors are then mapped to output text tokens. We use teacher-forcing during training and greedy autoregressive generation during evaluation.

We first check if the network could solve the tasks by producing the final yes/no responses immediately after the prompt. After training on 50,000 HS puzzles uniformly sampled from the full problem space, we find that these models only solve 87.8% of held-out puzzles, compared to the 99.9% of models trained to produce the full sequence of reasoning steps. Taking complete success at within-distribution generalization as a pre-requisite, we use the full sequences as exemplified in Table 1 in all remaining experiments, which focus on out of distribution generalization in our models.

Out-of-distribution generalization. We define within-distribution (WD) puzzles as those that conform to the restrictions used to construct the training set and out-of-distribution (OOD) as puzzles that do not conform to these restrictions. All models included in our analyses solved held-out WD puzzles with near perfect accuracy, so we only report OODG performance in our results for conciseness.

Based on the natural symmetries of Sudoku grids and the train/test split used in [10], we devised three distributional splitting schemes to probe for OODG. Our first condition, *Rows*, recognizes that the application of the HS, FH, or NS techniques is isometric relative to the row in question. Thus, a successful model of abstract relational reasoning should solve HS puzzles applied to any of the 6 rows, even when trained on a strict subset of the 6 rows. We train the models using HS puzzles applied to 4 of the 6 rows (i.e. the goal cell will only appear on these 4 rows), then test its OOD performance on the remaining 2 rows. In our second condition, *Columns*, we exclude all HS puzzles that are performed over columns from the training set, and test the models on these column puzzles. The abstract principle of process of elimination remains the same, and the main challenge is knowing which cells to iterate over. Our final condition, *Digits*, uses the fact that swapping digits only superficially changes the puzzle without affecting the underlying structure (see Figure 2. We train the model on puzzles with 4 of the 6 digits as the candidate digits and test the model on puzzles with the remaining 2.

We also had 3 conditions for how we trained the models. In the first training condition, HS Only, we included 110,000 HS puzzles as part of the training set and not the FH or NS puzzles, and trained the model for 70,000 gradient updates (each based on 192 examples). The second condition, *Simultaneous*, included 50,000 HS puzzles and 30,000 FH and NS puzzles each, and the model was trained on all 3 tasks simultaneously for 70,000 updates. The FH and NS puzzles were sampled completely at random without any systematic constraints. The last condition, *Curriculum*, uses the same materials as Simultaneous, but trains the model on the FH and NS puzzles for the first 20,000 updates, reaching ceiling performance, before training on all 3 tasks for 50,000 more updates.

Figure 1 summarizes the out-of-distribution generalization results in each condition. First, we compute the accuracy based on the final yes/no answer at the end of the output sequence, and we find that models trained only on HS puzzles struggle to exceed chance (50%), suggesting that the model has no inherent inductive bias towards generalizing in these dimensions. When trained with the FH and NS tasks, the model succeeds in generalizing to the held-out rows and columns to a high degree, especially when trained using the curriculum-based setup. We consider the strong generalization in the Columns condition (though somewhat less complete than in the Rows condition) to be an important finding, since the Columns condition contained no training on column-wise puzzles, while the Rows condition included 4 of the 6 rows. Our models generally failed to exceed chance on the held-out digit puzzles. The relational neural network [8] also fails to transfer to held-out digits, but humans who learn to solve HS puzzles with a restricted set of targets show no decrement when tested on held-out digits [10].

We also measure the accuracy for entire sequences, in which a problem is considered solved if the entire model output sequence matches the target sequence, including the intermediate steps. This not only magnifies the differences, but also indicates that the models that generalize successfully do so in its entire reasoning process, not just at the correct final output.

Although the Rows and Columns conditions successfully demonstrate out-of-distribution generalization, the Digits condition does not, as shown by the roughly 50% accuracy in Figure 1. This is surprising, since human participants show no change in performance when the digits in the grid are swapped [10]. Its peculiarity is magnified by the fact unlike the Rows and Columns conditions, full sequence accuracy is well above the floor even when the models are trained with only HS puzzles.

Digit generalization error analysis. Inspecting the model generated outputs, we find that the models correctly identify the relevant cells to iterate over, and the last line indicating the final answer is consistent with its intermediary outputs. In other words, the errors are made when determining whether or not the cells can contain the candidate digit. Moreover, these errors only occur at empty cells, not at cells already containing a digit.

Figure 2: Attentional maps. Within- and out-of-distribution puzzle with candidate digits 3 and 4 on top and bottom, respectively

To gain further insight, we analyze the attention maps generated by the transformer that indicate which information the model considers. We probe the model by taking held-out puzzles and rotating the digits in each grid such that 1 becomes 2, 2 becomes 3, 6 becomes 1, etc., thus producing 6 identical sets of puzzles that only differ by the digits involved, and each puzzle in the set has a unique candidate digit. We inspect how the model queries the grid as it produces the yes/no responses at the end of each substep in the HS problem by taking the maximum attention weight given for each cell from the last transformer layer. Figure 2 shows an example of one model's attentional map on the HS problem shown in Table 1 when considering whether the candidate digit can be placed at the cell on (2, 2). In the within-distribution problem, where 3 is the candidate digit, the model attends to and correctly identifies the 3 in the same row. In contrast, the model does not attend to the same position in the out-of-distribution puzzle where 4 is the candidate digit. This form of mis-attention is characteristic of the errors in the Digits condition.

The model apparently fails to transfer its success in the NS task to correctly perform the NS task when it is embedded within the HS task. We trace the source of this difficulty to a superficial difference between the NS and HS tasks. Unlike the FH task which is aligned token-for-token with the HS task, the NS task requires much fewer steps and has a shorter prompt compared to the HS and FH tasks. We test this hypothesis by padding the NS prompt with 10 null tokens so that the position of the "digit" token is aligned with the position in the HS and FH prompts, and by adding 4 extra "row r column c yes/no" lines in the target sequence with random coordinates to match the format of the

other two tasks. Training the models on all 3 tasks simultaneously on this *Padded* dataset allows them to reach near perfect generalization in the Digits condition. Interestingly, we find that adding the 4 extra lines but not the null-token padding does not help the model generalize at all, leaving the OODG accuracy near 50%. This suggests that the knowledge transfer from the NS to the HS tasks may be impeded by the model's over-reliance on the token position encoding used in transformers.

3 Discussion

We find, using carefully designed datasets utilizing isomorphic symmetries in Sudoku, that transformer-based models can generalize well to out-of-distribution puzzles in the Rows and Columns conditions when component subtasks span the full space of the data distribution, and fail to do so without this training on the component subtasks. These findings may be relevant to understanding the performance of large transformer-based models. While they may receive restricted experience with complex multi-step problems, their ability to solve new ones may depend in part on more complete coverage of component sub-problems in their training data. Future research should explore this hypothesis in larger and more naturalistic data sets. We did find that the models struggled to generalize to held-out digits, and found evidence that this failure is due to a failure to attend to within-distribution digits during crucial steps within the Hidden Single task. This is surprising in that the failure occurs during steps the model does succeed on in the component Naked Single task, and this is likely due to over-reliance on token position encoding. Future research that corrects this deficiency could have important implications for the success of larger models, perhaps helping them to achieve stronger performance in abstract reasoning than they have achieved thus far.

References

[1] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[2] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168, 2021.

[3] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

[4] Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *arXiv preprint arXiv:2206.14858*, 2022.

[5] Bertram Felgenhauer and Frazer Jarvis. Mathematics of sudoku i. *Mathematical Spectrum*, 39(1):15–22, 2006.

[6] Ed Russell and Frazer Jarvis. Mathematics of sudoku ii. *Mathematical Spectrum*, 39(2):54–58, 2006.

[7] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[8] Rasmus Palm, Ulrich Paquet, and Ole Winther. Recurrent relational networks. *Advances in neural information processing systems*, 31, 2018.

[9] NY Louis Lee, Geoffrey P Goodwin, and Philip N Johnson-Laird. The psychological puzzle of sudoku. *Thinking & Reasoning*, 14(4):342–364, 2008.

[10] Andrew Joohun Nam and James L McClelland. What underlies rapid learning and systematic generalization in humans. *arXiv preprint arXiv:2107.06994*, 2021.

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[12] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

4 Supplementary Materials

4.1 Training Details

The core of our model is a 3-layer transformer encoder [11] supported by embedding layers for grid digits, grid cell positions, and input text, and finally an output text decoder layer. The digits of the grid are encoded using a 256-dim embedding layer. The row and column coordinates are encoded by a 128-dim embedding layer and the resulting vectors are concatenated as a single 256-dim vector. The digit and coordinate vectors are summed to form a single 256-dim grid cell embedding. All text tokens are encoded by a 256-dim embedding layer and position information is added to the vectors using the sinusoidal positional encoding scheme introduced in [11]. The 36 grid cell vectors and all token vectors are passed to the transformer, which is composed of 3 encoder layers with 8 heads and 1024-dim feed-forward layers, and the output vectors are decoded using a linear layer to form the final logit values for the predicted output tokens.

During training, we use teacher-forcing to predict the next token at each sequence position and cross-entropy with the target sequence. We mask the loss so that in the loss is only applied after the 'digit' token in the hidden single and full house tasks, and after the column number token in the naked single tasks. The loss is computed using cross-entropy and the model is optimized using Adam [12] with a learning rate of 0.0001. When training with multiple tasks at once, we sum the losses in each batch from all the tasks before computing the gradient for backpropagation. We keep the same batch size of 192 samples for each task, regardless of how many tasks are used to train at once.

Although we provide the model with seed sequences as exemplified in Table 1 including the candidate digit at evaluation time, we train the model to also predict the candidate digit to evaluate whether how accurately could identify the correct digit with the prompt alone. During evaluation time, all output sequences are produced using greedy autoregressive generation.

4.2 Attention Map

To obtain the attention map as shown in Figure 2, we evaluated the model on the Hidden Single problem shown in Table 1. We generated 5 additional problems based on the problem in Table 1 by shifting the digit 1 to 5 times such that what was originally a 1 would be a 2 in the second problem, then a 3, and so on, yielding 6 puzzles that are identical in every way except the individual digits involved. For example, the top figure in Figure 2 shows the problem as is shown in Table 1, whereas the bottom figure shows the same puzzle with all digits incremented by 1, wrapping around 6 back to 1. In the bottom puzzle, the prompt would state "digit 4" to account for the increment.

After evaluating the model, we take the attentional weights from the final transformer layer where the input token was the second "2" from the line stating "row 2 column 2", since the output of this position would be a "yes" or a "no". To visualize the attention across all 8 heads, we take the maximum attention weight so that if a single head attended highly to the position, it would appear in the figure. Figure 2 only shows attention to the 36 cells in the grid for visualization purposes, though the model could and does attend to other tokens in the prompt and output sequence.