When Can Transformers Ground and Compose: Insights from Compositional Generalization Benchmarks

Ankur Sikarwar[† 1] **Arkil Patel**[† 2,3] **Navin Goyal**[4]

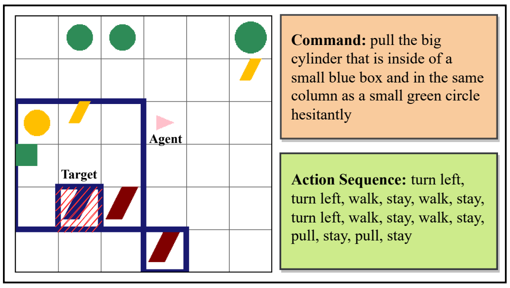
[1]I2R, A*STAR Singapore [2]Mila - Quebec AI Institute [3]McGill University
[4]Microsoft Research India
ankursikarwar.as@gmail.com
arkil.patel@gmail.com, navingo@microsoft.com

Abstract

Humans can reason compositionally whilst grounding language utterances to the real world. Recent benchmarks like ReaSCAN (Wu et al., 2021) use navigation tasks grounded in a grid world to assess whether neural models exhibit similar capabilities. In this work, we present a simple transformer-based model that outperforms specialized architectures on ReaSCAN and a modified version (Qiu et al., 2021) of gSCAN (Ruis et al., 2020). On analyzing the task, we find that identifying the target location in the grid world is the main challenge for the models. Furthermore, we show that a particular split in ReaSCAN, which tests depth generalization, is unfair. On an amended version of this split, we show that transformers can generalize to deeper input structures. Finally, we design a simpler grounded compositional generalization task, **RefEx**, to investigate how transformers reason compositionally. We show that a single self-attention layer with a single head generalizes to novel combinations of object attributes. Moreover, we derive a precise mathematical construction of the transformer's computations from the learned network. Overall, we provide valuable insights about the grounded compositional generalization task and the behaviour of transformers on it, which would be useful for researchers working in this area.

1 Introduction

Natural Languages are believed to be compositional (Partee et al., 1984), i.e., the meaning of an expression is determined by the meaning of its constituents and how they are combined. The field of *compositional generalization* seeks to understand whether neural models used for language processing exhibit compositional behaviour. In recent years, the field has received increased attention resulting in the development of many new benchmarks (Lake and Baroni, 2018; Kim and Linzen, 2020; Keysers et al., 2020) and approaches (Li et al., 2019; Lake, 2019; Chen et al., 2020; Liu et al., 2021) to solve them.



Command: pull the big cylinder that is inside of a small blue box and in the same column as a small green circle hesitantly

Action Sequence: turn left, turn left, walk, stay, walk, stay, turn left, walk, stay, walk, stay, pull, stay, pull, stay

Figure 1: An example from the ReaSCAN dataset.

Natural language utterances are also grounded to the real world. To encourage development of systems that are both *compositional* and *grounded*, Ruis et al. (2020) created the gSCAN dataset. Recently, Qiu et al. (2021) proposed the GSRR dataset[1] and Wu et al. (2021) proposed the ReaSCAN dataset to address certain limitations in gSCAN. These tasks consist of navigation commands grounded in a 2D grid world containing an agent and multiple objects with different visual attributes. Given a command and grid world, a model needs to output the sequence of actions for the agent to execute. Fig. 1 shows an example from ReaSCAN. The difficulty of the task lies in generalizing to out-of-distribution splits that are formed by systematically holding out particular compositions of object attributes and command structures from train set. Heinze-Deml and Bouchacourt (2020); Kuo et al. (2021) developed specialized architectures for gSCAN that are either difficult to adapt to other problems or require extra supervision.

Contributions. Our goal is to better understand these grounded compositional generalization tasks and design *generic* ML models to solve them. Our contributions include:

[†] Work done at Microsoft Research India.

[1] They proposed new test splits for gSCAN, which we call Grounded Spatial Relation Reasoning (GSRR).

Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 648–669
December 7-11, 2022 ©2022 Association for Computational Linguistics

`Simple:`
Walk to the small red square.
`1-relative-clause:`
Pull the blue circle **that is** in the same row as the small green square.
`2-relative-clause:`
Push the small blue cylinder **that is** in the same column as the big green circle **and** the red square.

Table 1: Different types of ReaSCAN commands.

Split	Held-out Examples
A1	*yellow squares* referred with color and shape
A2	*red squares* as target
A3	*small cylinders* referred with size and shape
B1	*small red circle* and *big blue square* co-occur
B2	*same size as* and *inside of* relations co-occur
C1	additional conjunction clause added to `2-relative-clause` commands
C2	`2-relative-clause` command with *that is* instead of *and*

Table 2: Compositional splits in ReaSCAN.

(i) We propose the **Gro**unded **Co**mpositional **T**ransformer (GroCoT), which was created by making simple and well-motivated modifications to a multi-modal transformer model (Qiu et al., 2021). GroCoT achieves state-of-the-art performances on both, GSRR and ReaSCAN.[2] Our results clearly show that simple transformer-based models generalize well on these tasks.

(ii) We design a series of experiments to understand the underlying challenges in these tasks. We show that identifying the target location, rather than sequence generation, is the main difficulty. We also demonstrate that the split, testing depth generalization in ReaSCAN is unfair in that the training data does not provide the models with sufficient information to correctly choose among competing hypotheses. On experimenting with a modified training distribution, we show that simple transformer-based models can successfully generalize to commands with greater depths.

(iii) We examine why transformers are so successful at generalizing compositionally on these tasks. To this end, we introduce a new task called **RefEx** ('Referring Expressions'), which provides a simpler setting isolating some of the main features of ReaSCAN. We find that a 1-layer, 1-head attention-only transformer is capable of grounding and generalizing to novel compositions of multiple visual attributes; moreover, it admits a complete interpretation of the computations. RefEx also allows easier probing and leads us to identify and solve an overfitting issue with transformers on a particular ReaSCAN split.

2 Background

We focus on the gSCAN (Ruis et al., 2020), GSRR (Qiu et al., 2021) and ReaSCAN (Wu et al., 2021) datasets. A model, provided with a natural language command, is tasked with generating a sequence of actions to navigate an agent in a 2D grid world populated with objects. Below, we shall explain the setting of the ReaSCAN task in detail. More information about other datasets is provided in Appendix C.

Each example consists of a $d \times d$ grid world ($d = 6$), a natural language command and the corresponding output sequence. Each cell in the grid world is described by a c-dimensional vector that concatenates one-hot encodings for the three object attributes, color $\mathcal{C} = \{\mathsf{red}, \mathsf{green}, \mathsf{blue}, \mathsf{yellow}\}$, shape $\mathcal{S} = \{\mathsf{circle}, \mathsf{square}, \mathsf{cylinder}, \mathsf{box}\}$, and size $\mathcal{D} = \{1, 2, 3, 4\}$ along with information about agent orientation $\mathcal{O} = \{\mathsf{left}, \mathsf{right}, \mathsf{up}, \mathsf{down}\}$ and agent presence $\mathcal{B} = \{\mathsf{yes/no}\}$. Hence, the entire grid world is represented as a tensor $\mathbf{W} \in \mathbb{R}^{d \times d \times c}$. The natural language command $x := (x_1, x_2, \ldots, x_n)$ is generated using a context-free grammar (CFG), which is described in Appendix C.1. ReaSCAN has three types of input commands which we illustrate in Table 1. The output sequence $y := (y_1, y_2, \ldots, y_m)$ is made up of a finite set of action tokens $\mathcal{A} = \{\mathsf{walk}, \mathsf{push}, \mathsf{pull}, \mathsf{stay}, \mathsf{turn\ left}, \mathsf{turn\ right}\}$.

The main challenge of the task is generalizing on the specially designed test splits that consist of various types of examples systematically held-out from the train set as shown in Table 2 (more details in Appendix C.1).

The results of various previously proposed methods are shown in Table 3 for GSRR, Table 4 for ReaSCAN, and Table 12 for gSCAN. Qiu et al. (2021) outperformed all previous methods (Gao et al., 2020; Kuo et al., 2021; Heinze-Deml and Bouchacourt, 2020) on gSCAN and GSRR. Hence, for ReaSCAN, we don't re-implement those methods as baselines; rather, we compare directly against Qiu et al. (2021).

[2] We make our source code and data available at: https://github.com/ankursikarwar/Grounded-Compositional-Generalization.

Model	Random (I)	Comp. Average	II	III	IV	V	VI
Multimodal LSTM (Wu et al., 2021)	86.5	58.9	40.1	86.1	5.5	81.4	81.8
Multimodal Transformer (Qiu et al., 2021)	94.7	63.5	64.4	94.9	49.6	59.3	49.5
GroCoT (ours)	**99.9**	**98.8**	**98.6**	**99.9**	**99.7**	**99.5**	**96.5**

Table 3: Performance of GroCoT on GSRR (Qiu et al., 2021) in comparison to baselines and previous approaches.

Model	Average	A1	A2	A3	B1	B2	C1	C2
Multimodal LSTM (Wu et al., 2021)	40.4	50.4	14.7	50.9	52.2	39.4	49.7	25.7
GCN-LSTM (Gao et al., 2020)	60.5	92.3	42.1	87.5	69.7	52.8	57.0	22.1
Multimodal Transformer (Qiu et al., 2021)	69.9	96.7	58.9	93.3	79.8	59.3	75.9	25.5
GroCoT w/ vanilla self-attention	80.8	99.2	88.1	98.7	**94.6**	**86.4**	75.3	23.4
GroCoT (ours)	**82.2**	**99.6**	**93.1**	**98.9**	93.9	86.0	**76.3**	**27.3**

Table 4: Performance of GroCoT on ReaSCAN (Wu et al., 2021) in comparison to baselines and previous approaches.

3 Our Approach

We start with the multimodal transformer model as used in Qiu et al. (2021). This model, hereafter called the *base model*, follows encoder-decoder structure Vaswani et al. (2017) and uses cross-modal attention in the encoder.

Encoder maps world state $\mathbf{W} \in \mathbb{R}^{d\times d\times c}$ to visual representation $\mathbf{H}^v$ through multi-scale CNNs followed by linear layers. The command tokens $x := (x_1, x_2, \ldots, x_n)$ are encoded into embeddings $\mathbf{H}^l = \{\mathbf{h}_1^l, \mathbf{h}_2^l, \ldots, \mathbf{h}_n^l\}$. These are passed through N transformer blocks, each consisting of two parallel multi-head attention blocks (one for vision and one for language modality), with representation of one modality passed as key and value to the attention block of the other modality.

Decoder consists of N stacked blocks similar to the decoder in Vaswani et al. (2017). Each block contains one self-attention block and one multi-head attention block over the contextual representation $\mathbf{H}_c = [\mathbf{H}_l; \mathbf{H}_v]$ of the encoder.

Below, we describe the modifications we make to this base architecture to create GroCoT. Implementation details are provided in Appendix B.

Improving Spatial Representation. For this task, models need to perform spatial reasoning between objects that may possibly be very far from each other in the grid world. The base model (Qiu et al., 2021) employed a multi-scale CNN to encode the world state $\mathbf{W}$ before feeding it to the transformer. However, CNNs, without the presence of large filters (i.e., large receptive fields), are inept at understanding the spatial relationships between parts of the image that are not in immediate vicinity. To address this limitation, instead of passing the world state tensor $\mathbf{W}$ through a multi-scale CNN, we propose tokenizing the grid cells and projecting them to a higher dimension $\mathbf{W}^v = \{\mathbf{w}_1^v, \mathbf{w}_2^v, \ldots, \mathbf{w}_{d\times d}^v\}, \mathbf{w}_i^v \in \mathbb{R}^{d_\text{model}}$. In line with Lu et al. (2019), we separately encode the spatial information of grid cells in a 2D vector, where the first dimension holds the row value and the second one holds the column value. We project these spatial encodings to a higher dimension $\mathbf{S}^v = \{\mathbf{s}_1^v, \mathbf{s}_2^v, \ldots, \mathbf{s}_{d\times d}^v\}, \mathbf{s}_i^v \in \mathbb{R}^{d_\text{model}}$ and add them to their corresponding grid cell representations to obtain the final grid cell embedding input to the transformer $\mathbf{H}^v = \{\mathbf{h}_1^v, \mathbf{h}_2^v, \ldots, \mathbf{h}_{d\times d}^v\}, \mathbf{h}_i^v = \mathbf{w}_i^v + \mathbf{s}_i^v \in \mathbb{R}^{d_\text{model}}$.

Interleaving Self-Attention. The base model uses cross-modal attention in all encoder layers. While this facilitates grounding of semantic information across both modalities, we believe this method to be inefficient. We know that different layers in both vision transformers and language transformers encode different levels of semantic knowledge (Dosovitskiy et al., 2021; Raghu et al., 2021; Jawahar et al., 2019). To allow efficient grounding, we want both visual and language modality streams to develop their own representations before synchronizing them with each other via cross-modal attention. Hence, we propose interleaving self-attention layers between co-attention layers to allow intra-modal interaction within each stream before cross-modal interaction.

Modified World State Encoding. In ReaSCAN, for particular examples where another object is present in the top left corner of a box object, the grid cell embedding corresponding to that corner is calculated by adding up the vector encodings corresponding to the object and the box (Wu et al.,

ISR	ISA	EM	AVERAGE	A1	A2	A3	B1	B2	C1	C2
✗	✗	✗	69.9	96.7	58.9	93.3	79.8	59.3	75.9	25.5
✓	✗	✗	70.7	96.4	75.2	93.8	78.2	57.9	71.7	21.9
✗	✓	✗	77.7	99.1	87.6	98.4	89.2	67.7	**79.3**	22.3
✗	✗	✓	73.6	97.4	70.7	94.3	83.2	66.4	76.7	26.3
✓	✓	✗	79.0	99.0	90.9	98.2	88.3	72.7	77.0	26.9
✗	✓	✓	77.8	98.5	79.6	97.8	90.8	78.9	78.6	20.8
✓	✗	✓	73.9	97.7	76.7	96.0	82.1	64.8	75.1	25.1
✓	✓	✓	**82.2**	**99.6**	**93.1**	**98.9**	**93.9**	**86.0**	76.3	**27.3**

Table 5: Ablation study for GroCoT on ReaSCAN (Wu et al., 2021). The results show that all our modifications are necessary to achieve best performance. ISR, ISA, and EM stand for Improved Spatial Representation, Interleaving Self-Attention, and Embedding Modification, respectively.

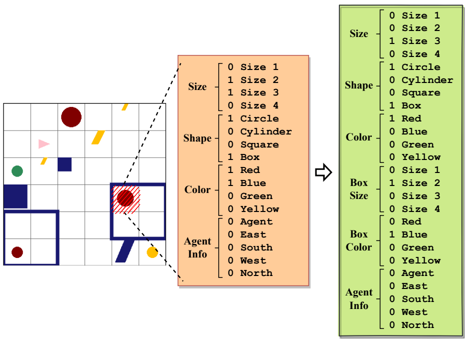


Figure 2: Modified Grid Cell Encoding

2021). However, this design is inherently flawed because the attributes of the two objects cannot be disambiguated from the sum of their individual encodings. This issue causes models to fail in such examples. To handle such cases, we propose using a higher dimensional grid cell embedding (see Fig. 2) to represent color and size properties of the box separately from other objects.

Discussion. The results are provided in Table 3 for GSRR and Table 4 for ReaSCAN. We also provide exhaustive ablations of our approach on ReaSCAN in Table 5. On both datasets, GroCoT outperforms all specialized architectures. From the ablation study on ReaSCAN, we observe that both, improved spatial representation and interleaved self-attention, lead to significant improvements. The modified embedding structure additionally helps when examples contain box objects (see improvement in ReaSCAN B2 split). Our model also saturates performance on most splits in gSCAN (see Table 12). We also evaluated the effect of using vanilla self-attention (as used in the original Transformer (Vaswani et al., 2017)) in GroCoT[3] on ReaSCAN and found that it achieves surprisingly high accuracies (see Table 4). Our hypothesis is that vanilla self-attention facilitates individual processing of both modalities similar to our interleaving self-attention approach, and hence it does not hurt the model performance significantly.

Overall, our results show that *a simple transformer-based model is capable of generalizing compositionally on most of the challenges proposed by gSCAN, GSRR and ReaSCAN.* Instead of presenting GroCoT as a broadly applicable method solving compositional generalization, we only wish to establish that such simple transformer-based models can exhibit strong compositional generalization capabilities and serve as powerful baselines.

4 Analyzing the Grounded Compositional Generalization Tasks

4.1 Target Identification vs Navigation: What is the Challenge?

In order to solve these tasks, a model needs to perform two subtasks: (1) identify the target location by composing the words and reasoning about the relative clauses, and (2) navigate the agent in the grid world by generating the right set of output tokens. To understand why models fail and how to improve them, we need to pinpoint where the main difficulty in the task lies. Below we describe a set of experiments that demonstrate that target identification is the main challenge in ReaSCAN rather than navigation or sequence generation.[4] We show this for the gSCAN dataset in Appendix E.1.3.

Target Identification from Encoder Representations. We train a linear layer on top of the last layer of learned encoder representations of the best-performing model to perform target identification.[5]

[3] Note that our other proposed modifications (improved spatial representation and embedding modification) are still being applied.

[4] Studies with similar objectives have also been carried out by past work. We explain their limitations and compare against them in Appendix E.1.1.

[5] We also experimented with predicting the target location

	Average	A1	A2	A3	B1	B2	C1	C2
Target Identification accuracy	78.5	96.4	85.7	95.4	90.4	83.5	70.2	27.7
Error overlap with ReaSCAN model	88.3	89.5	87.9	83.5	87.4	86.7	88.5	94.7
ReaSCAN accuracy w/ gold target locations	99.9	99.9	99.9	99.9	99.9	99.9	99.9	99.9

Table 6: Experimental results for GroCoT to understand the performance bottleneck in ReaSCAN.

ISR	ISA	EM	Average	A1	A2	A3	B1	B2	C1	C2
✗	✗	✗	69.8	94.3	59.4	91.0	79.9	64.3	74.4	25.1
✓	✗	✗	71.9	95.4	87.0	94.1	75.7	56.0	71.6	23.5
✗	✓	✗	77.7	96.0	84.3	95.5	90.5	73.4	78.2	25.7
✗	✗	✓	77.0	95.5	76.3	93.4	87.8	81.2	75.8	**29.1**
✓	✓	✗	74.6	95.6	85.4	95.1	84.7	69.8	71.4	20.0
✗	✓	✓	**81.1**	**96.5**	**89.5**	**96.5**	**91.9**	**88.1**	**80.7**	24.3
✓	✗	✓	72.7	95.7	71.7	93.7	82.0	73.2	71.4	21.2
✓	✓	✓	78.5	96.4	85.7	95.4	90.4	83.5	70.2	27.7

Table 7: Target Identification accuracy for different ablations on GroCoT when tested on ReaSCAN (Wu et al., 2021). ISR, ISA, and EM stand for Improved Spatial Representation, Interleaving Self-Attention, and Embedding Modification, respectively.

We model the task as a 36-way classification problem, where each grid location is treated as a distinct class. We train the model over all ReaSCAN examples with the ground truth target locations. Note that we only update the weights of the linear layer; the parameters of the encoder are kept frozen. We then test this model's target identification abilities over the systematic generalization splits.

The first row of Table 6 shows the performance of the model on target identification. We see the same trend in performance across all splits as we saw for the full model on ReaSCAN (see Table 4). This indicates that target identification might be the main difficulty for the model. To illustrate more concretely, we calculate the overlap of errors between the target identification model and the ReaSCAN model. As can be seen from the second row in Table 6, for each split, out of all the examples where the ReaSCAN model failed, the percentage of examples where the target identification model also failed is extremely high.

We also provide exhaustive ablations for this experiment in Table 7. Our proposed modifications do indeed enable better target identification. However, there might be other minor aspects of the problem that are tackled by our modifications (model with ISR performs better overall on ReaSCAN but is not the best on target identification). Also note that these experiments are performed without the decoder, which is essential in solving ReaSCAN.

Sequence Generation from Gold Target Locations. We provide the model with gold target locations when training it on the ReaSCAN training set. We enumerate all the 36 grid cell locations and simply append gridnum to the end of the natural language command where gridnum $\in \{1, 2, ..., 36\}$, depending on the ground-truth target location for a particular example. The results are provided in the last row of Table 6. Clearly, the model is able to generalize almost perfectly when provided with the ground-truth target locations. This shows that if the target is identified, the model has no difficulty in navigating the agent towards it.

In this section, with comprehensive empirical evidence, we showed that *models are highly competent at agent navigation* and that *the chief difficulty lies in identifying the target location.*

4.2 Issues in ReaSCAN Test Set Design

Compositional generalization setups are used to assess specific capabilities of models. However, if the train-test splits within these setups are not carefully created, then the experimental results may lead to false conclusions (Patel et al., 2022). In this section, we show that the C2 test set of ReaSCAN is unfair because of lack of necessary information in the train set. We then propose a correction in the train-test setup that allows us to fairly evaluate the depth generalization capabilities of models.

C2 Test Set is Unfair. The train set of ReaSCAN consists of commands with different structures as shown in Table 1. The C2 test set is

from earlier encoder representations and random vectors to serve as baselines. These results are provided in Figure 10.

made up of commands with the other type of `2-relative-clause` structure (e.g., "walk to the red square *that is* in the same row as the blue cylinder *that is* in the same column as a green circle"). This split tests whether a model is able to perform recursion to higher depths. It is clear that the only difference between the `2-relative-clause` commands in the train set and the C2 test set is that the 'and' connecting the last two clauses is replaced with 'that is'. Hence it is crucial for the model to understand the difference between these terms to successfully generalize on the C2 test set. However, based on the train set, they both perform the same role: they act as a connector between two clauses in the command where the target needs to be identified based on the attributes in the first clause after satisfying the constraint of the following clause. We explain this more intuitively in Appendix E.2.1. To illustrate this empirically, we show that the average consistency between model predictions before and after replacing all 'and' with 'that is' in ReaSCAN test sets is 93.5.[6] Hence, *the train set of ReaSCAN is insufficient for the model to disambiguate between 'and' and 'that is', thereby rendering the C2 generalization task unfair.*

Model Learns a Reasonable Alternate Hypothesis for C2. We hypothesize that for the commands in the C2 split, the model treats the second 'that is' as if it were 'and', similar to the `2-relative-clause` commands it has seen in the train set. We consider this model behavior to be reasonable, since this hypothesis is consistent with the train set. To verify this empirically, we create a new set of examples, called C2-`alt`, by replacing the second 'that is' with an 'and' in all examples in the C2 test set. The model's predictions for C2 matched those for C2-`alt` 93% of the time[7], clearly validating our hypothesis.

Transformers Generalize to Higher Depths of Relative Clauses. Since we showed that the C2 test set is unfair, we designed a new, fairer test split to check generalization capability of models to higher depths. We include commands up to depth 2 (i.e., the type of commands in the C2 test set) in the train set and test on commands of depth 3. We call this new test set as C2-`deeper`.[8] By including commands up to depth 2, we alleviate the issue of the model not being able to disambiguate between 'that is' and 'and'. Our best model achieved 85.6% accuracy on C2-`deeper`. This is very surprising (since transformers have been known to struggle at depth recursion (Kim and Linzen, 2020)) and clearly shows that *the multimodal transformer model is capable of generalizing systematically to higher depths* in this setting. This also re-affirms our claim that the original C2 test set was unfair.

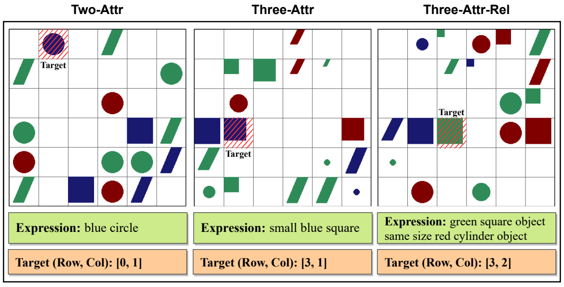

Two-Attr
Three-Attr
Three-Attr-Rel
Target
Target
Target
Expression: blue circle
Expression: small blue square
Expression: green square object same size red cylinder object
Target (Row, Col): [0, 1]
Target (Row, Col): [3, 1]
Target (Row, Col): [3, 2]

Figure 3: Examples from our RefEx dataset.

5 RefEx: Understanding How Transformers Ground and Compose

We wish to understand how transformers succeed on grounded compositional generalization tasks. However, the complexity of both, the ReaSCAN task, and the multi-modal transformer, makes it difficult to interpret the model. Hence, we design a new task, **RefEx**, based on the target identification subtask of ReaSCAN.[9] In the following sections, we show that even a one-layer, one-head attention-only model can successfully ground and compose multiple object attributes in RefEx. We then give a precise construction of the model, which demonstrates the detailed computations corresponding to grounding and composition.

5.1 Task Setup and Test Splits

Given a command that refers to a unique target object in the accompanying grid world, a model needs to identify the target location. Compared to ReaSCAN, etc., we get rid of subtasks like path planning and action sequence generation, and focus only on target identification.

We design three variants of the RefEx task with different command structures varying in difficulty:
(i) `two-attr:=$COL $SHP`. Model needs to ground and compose *color* and *shape* attributes.
(ii) `three-attr:=$SIZ $COL $SHP`. Model needs

[6] Detailed results can be found in Table 14.

[7] Note that the model prediction for C2-`alt` matches the corresponding ground-truth for C2-`alt` 99% of the time, showing that the model correctly handles those type of commands.

[8] Details of this setup are provided in Appendix E.2.2.

[9] We discuss how RefEx differs from similar synthetic benchmarks in Appendix F.1.

SPLIT	HELD-OUT EXAMPLES
A1	*green squares* as targets
A2	*red circle* as targets or distractors
A3[10]	*green circles* of size 2, referred with "small"
A4[10]	command is "small blue cylinder"

Table 8: Held-out examples in RefEx's compositional splits. Detailed descriptions are provided in Appendix F.2.1.

VARIANT	LAYERS	R	A1[11]	A2	A3	A4
`two-attr`	1	100	100	100	-	-
`three-attr`	1	100	100	100	100	100
`three-attr-rel`	1	78.8	31.9	33.5	-	-
	2	99.7	99.4	98.8	-	-

Table 9: Performance of attention-only transformer (with single attention head in each layer) on the RefEx task. R stands for random.

to additionally handle the *size* attribute, which requires relative reasoning.

(iii) `three-attr-rel:=$OBJ $REL $OBJ`. Model effectively needs to perform two `three-attr` subtasks sequentially based on the relation between the referent and target objects.

Here, `$COL` $\in$ $\{\text{red}, \text{green}, \text{blue}\}$, `$SHP` $\in$ $\{\text{square}, \text{circle}, \text{cylinder}\}$, `$SIZ` $\in$ $\{\text{small}, \text{big}\}$, `$OBJ:= $SIZ? $COL? $SHP?`, and `$REL` $\in$ $\{\text{same size}, \text{same color}, \text{same shape}\}$. Figure 3 shows examples from all three variants. Additionally, we create four compositional generalization test splits as described in Table 8.

5.2 Model

We consider attention-only transformers (with residual connections) with two layers or less. We use natural sparse embedding matrices (see Fig. 14, 15, 16) to represent the command and world state. The input sequence length $n = 2 + 36$ for `two-attr` where 2 and 36 correspond to the number of command and grid world tokens, respectively. The output representations corresponding to the grid world tokens are mapped to logits by taking element-wise sum followed by softmax for 36-way classification (each class represents a unique grid location). See Appendix F.2 for more details.

5.3 Results and Discussion

The performance of the model described above on the RefEx task is shown in Table 9.

Efficacy of Self-attention Layers. For `two-attr`, we found it surprising to see that a one-layer, one-head attention-only transformer can successfully ground and compose the attributes for correct target identification. Moreover, the model generalizes to novel compositions of the attributes as can be seen from its performance on the compositional splits.

In `three-attr`, which is more difficult than `two-attr`, surprisingly, a one-layer, one-head model can ground and compose three different attributes, including *size*, which requires complex relative reasoning. Finally, in `three-attr-rel`, we find that at least two layers are required to solve the task. This makes intuitive sense: each layer will solve one `three-attr` subtask to identify the referent or target object.

From RefEx to ReaSCAN. Inspired by these results, we evaluate attention-only transformers on ReaSCAN target identification. Examples in ReaSCAN can contain up to three referring expression subtasks. Therefore, based on our intuition above, the model's performance should saturate after 3 layers. We show these results in Fig. 12. Moreover, since the design of splits in RefEx is similar to that of ReaSCAN, we were able to derive useful insights from models trained on RefEx in order to improve the performance on ReaSCAN A2 split. See Appendix F.3 for details.

5.3.1 Interpreting how Transformers Ground and Compose

We completely describe how an attention-only transformer with one attention layer and one head solves `two-attr` (`three-attr` and `three-attr-rel` are similar but more complex). Let's first recall how the one-layer, one-head model works. We denote the input token embeddings by $\mathbf{x}_1, \ldots, \mathbf{x}_n$ and the final output embeddings by $\mathbf{r}_1, \ldots, \mathbf{r}_n$. Let $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V$ be the parameter matrices for queries, keys, and values; we can then define the query, key and value vectors for $i \in [n]$ by $\mathbf{q}_i = \mathbf{W}_Q \mathbf{x}_i$, $\mathbf{k}_i = \mathbf{W}_K \mathbf{x}_i$, and $\mathbf{v}_i = \mathbf{W}_V \mathbf{x}_i$. For each $i \in [n]$ we compute the output of attention block as $\tilde{\mathbf{r}}_i = \mathbf{W}_o \sum_{j=1}^{n} \alpha_{i,j} \mathbf{v}_j$, where the attention scores are given by $(\alpha_{i,1}, \ldots, \alpha_{i,n}) = \mathrm{softmax}(\langle \mathbf{q}_i, \mathbf{k}_1 \rangle, \ldots, \langle \mathbf{q}_i, \mathbf{k}_n \rangle)$. The residual connection then gives $\mathbf{r}_i = \tilde{\mathbf{r}}_i + \mathbf{x}_i$. As described in Section 5.2, in our model, the final grid cell containing the target is chosen by applying softmax to the logits corresponding to 36 grid world tokens, $L_{n-36}, \ldots, L_n$. Let $\mathbf{1}$ be the all-ones vector; for `two-attr`, we have $\langle \mathbf{1}, \mathbf{x}_i \rangle = 2$ when i corre-

[10] Only for the `three-attr` variant.

[11] Result on modified training distribution. See Section F.3.

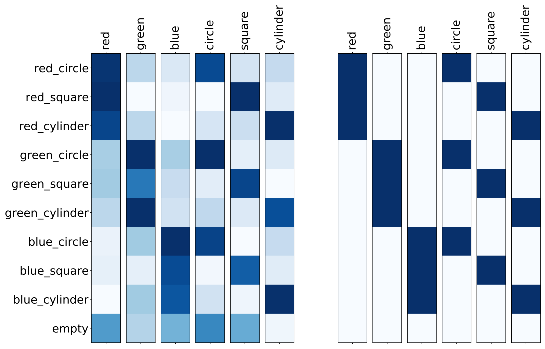
sponds to grid world tokens (see embedding matrix in Figure 14). Now, we show the computation of logit L_i where i corresponds to grid world tokens.

$$\begin{aligned} L_i &= \langle \mathbf{1}, \mathbf{r}_i \rangle = \langle \mathbf{1}, \mathbf{x}_i \rangle + \langle \mathbf{1}, \tilde{\mathbf{r}}_i \rangle \\ &= 2 + \langle \mathbf{1}, \mathbf{W}_o \sum_{j=1}^{n} \alpha_{i,j} \mathbf{v}_j \rangle \\ &= 2 + \sum_{j=1}^{n} \alpha_{i,j} \langle \mathbf{1}, \mathbf{W}_o \mathbf{v}_j \rangle \\ &= 2 + \sum_{j=1}^{n} \alpha_{i,j} \mathbf{s}_j \end{aligned}$$

We now qualitatively illustrate on a specific example of `two-attr` and show how the learned parameters ($\mathbf{M}_{\text{Learned}}$ in Figure 4)) lead to the correct target prediction. Note that the matrix $\mathbf{M} \in \mathbb{R}^{n_{\text{vocab}} \times n_{\text{vocab}}}$ here contains the dot product of query and key vectors of all possible pairs of tokens in the vocabulary. The rows in $\mathbf{M}$ correspond to queries while the columns correspond to keys. Let the command tokens be ⟨red⟩, ⟨cylinder⟩. We now show that the logit for ⟨red cylinder⟩ is the maximum among all possible values for grid tokens. In the learned model, we observe that when i and j correspond to grid world tokens either $\alpha_{i,j}$ or $\mathbf{s}_j$ is very small :

$$\mathbf{L}_i \approx 2 + \sum_{j \in C} \alpha_{i,j} \mathbf{s}_j,$$

where C is the set of indices of command tokens in our example, i.e., $C = \{1, 2\}$. When there is a match in an attribute in the grid world token i (say ⟨red circle⟩) and a command token j (say, ⟨red⟩), then the corresponding summand in the above sum, i.e. $\alpha_{i,j}\mathbf{s}_j$ is large, and when there is no match (e.g., when the command token j is changed to green) then it is small. Therefore, in our example, the logit computation for the token ⟨red cylinder⟩ has two large summands and corresponds to a *full match*, whereas for tokens like ⟨blue cylinder⟩ and ⟨red circle⟩, there is only one large summand. This corresponds to *partial match*. Finally, for a token like ⟨green square⟩, there is *no match* i.e. none of the summand, in this case, is large. While the above argument is qualitative, by looking at the entries of $\mathbf{M}_{\text{Learned}}$ it can be made quantitative; $\mathbf{M}_{\text{Learned}}$ led us to construct $\mathbf{M}_{\text{Construct}}$ (Fig. 4(b)) which makes the computations transparent while staying faithful to the learned model. Looking along the columns of the matrix corresponding to tokens ⟨red⟩ and ⟨cylinder⟩, we can see that the row corresponding to ⟨red cylinder⟩ grid world token has two "dark" cells (*full match*), while rows corresponding to ⟨blue cylinder⟩, ⟨red circle⟩ and ⟨green square⟩ have at most one "dark" cell (i.e. *partial match* or *no match*). Thus the grid cell corresponding to ⟨red cylinder⟩ will be the model's output. *Full match* here corresponds to the idea that both visual attributes ⟨red⟩ and ⟨cylinder⟩ mentioned in the command, were successfully grounded to the grid world token ⟨red cylinder⟩, and the composition of these two successful groundings contributed to two large summands in the final logit computation.



(a) Learned Model (b) Our Construction

Figure 4: (a) Portion of matrix $\mathbf{M}_{\text{Learned}}$ for the attention-only transformer trained on the `two-attr` variant and (b) Portion from $\mathbf{M}_{\text{Construct}}$ matrix (right side). Darker grid cells correspond to higher values in the matrix. Refer to Figure 17 and 18 for full matrices.

Finally, empirically our constructions attain perfect accuracies across all splits. The matrices of our construction for `three-attr` are in Figure 20 and 22.

6 Related Works

Compositional Generalization. Modern deep learning models perform extremely well on in-distribution test sets. However, unlike humans, they fail at generalizing compositionally (Lake and Baroni, 2018; Kim and Linzen, 2020; Keysers et al., 2020). Recent works have investigated the compositional generalization abilities of models in grounded setups using datasets such as CLEVR (Johnson et al., 2017), CLOSURE (de Vries et al., 2019), and gSCAN (Ruis et al., 2020). In this work, we focus on the task setup of gSCAN, and additionally work with GSRR (Qiu et al., 2021) and ReaSCAN (Wu et al., 2021). Both these works propose new challenging splits for gSCAN. Prior works have proposed many different specialized methods for solving gSCAN (Gao et al., 2020; Kuo et al., 2021; Heinze-Deml and Bouchacourt, 2020). Similar to Csordás et al. (2021); Patel et al. (2022), in this work, we show that even simple transformer-based models, with minor modifications to the architecture or training data distribution, perform well on the task and serve as strong baselines for future work.

Probing and Interpreting Models. In this work, we used a linear probe (Belinkov, 2022) to analyze the target identification abilities of models. Similar to our work, Weiss et al. (2021); Elhage et al. (2021); Kobayashi et al. (2020) and many others also attempt to explain the inner workings of transformers, possibly for non-synthetic problems. Unlike most of these works, however, our construction essentially *completely* describes the computations of the learned models. To the best of our knowledge, we are the first to study *how* self-attention facilitates compositional generalization in grounded environments.

7 Conclusion and Future Work

Recent benchmarks like gSCAN and ReaSCAN test grounded compositional generalization abilities of ML models. In this work, we identify key modifications in multimodal transformers that improve compositional generalization on these benchmarks. With a battery of probing experiments, we found that identifying the target location is the main challenge for the models. Additionally, we showed that a particular test set in ReaSCAN is *unfair* and proposed a modified train-test split in its stead. Finally, we designed a new task, **RefEx**, to study grounding and composition in attention-only transformers. We showed the efficacy of *single* self-attention layer with *single* head in successfully grounding and composing multiple visual attributes in a grid world environment. From the learned models, we derived an explicit and interpretable construction that captures the model's behavior and completely describes the detailed computations corresponding to grounding and composition.

While our focus has been on *tabula rasa* models, it is also of interest to see if pretraining on large datasets enables good performance on the considered benchmarks. Our preliminary investigations on GPT-3 and Codex (Appendix D) suggest that these text-based models have some way to go; more thorough investigation is left for the future.

We expect future work to address the current limitations of models on action sequence side compositional generalization, i.e., generalizing to novel combinations of action tokens. Moreover, our results indicate that designing compositional generalization splits can be surprisingly subtle and require careful scrutiny. Finally, grounded compositional generalization benchmarks should also target more realistic setups with natural images in the future.

8 Limitations

Our proposed approach fails on some gSCAN splits, specifically D, G, and H. These splits are designed to test output sequence-side systematic generalization capabilities of the model. In the future, we intend to extend our model by making architectural changes on the decoder side in order to tackle these splits.

On ReaSCAN, our approach achieves 86.1% on the B2 split and 76.3% on the C1 split, which is relatively less compared to the performance on other splits, suggesting room for further improvement. We expect that the C1 split also has similar issues like the C2 split (see section 4.2), although we haven't yet succeeded at concretely identifying these issues.

In line with previous models, our model also uses grid cell encodings which are very simple and explicitly represent different attributes of the world objects. However, natural images are high-dimensional and contain entangled representations of object attributes. Ideally, we would like to evaluate the compositional generalization abilities of models in a real-world setting using natural images since that has direct applications.

Our constructions for attention-only transformers in Section 5 are given only for the RefEx task which is a relatively simpler task than ReaSCAN. We plan to give similar constructions for more complex tasks than RefEx in the future.

Acknowledgements

We thank the anonymous reviewers for their constructive comments and suggestions. We would also like to thank Satwik Bhattamishra for reviewing an early draft of the paper and providing valuable feedback. We are grateful to our colleagues at Microsoft Research India for engaging in useful discussions and constantly supporting us throughout this work.

References

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Neural module networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 39–48.

Yonatan Belinkov. 2022. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Xinyun Chen, Chen Liang, Adams Wei Yu, Dawn Song, and Denny Zhou. 2020. Compositional generalization via neural-symbolic stack machines. In *Advances in Neural Information Processing Systems*, volume 33, pages 1690–1701. Curran Associates, Inc.

Róbert Csordás, Kazuki Irie, and Juergen Schmidhuber. 2021. The devil is in the detail: Simple tricks improve systematic generalization of transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 619–634, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Harm de Vries, Dzmitry Bahdanau, Shikhar Murty, Aaron C. Courville, and Philippe Beaudoin. 2019. CLOSURE: assessing systematic generalization of CLEVR models. In *Visually Grounded Interaction and Language (ViGIL), NeurIPS 2019 Workshop, Vancouver, Canada, December 13, 2019*.

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

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2021/framework/index.html.

Tong Gao, Qi Huang, and Raymond Mooney. 2020. Systematic generalization on gSCAN with language conditioned embedding. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 491–503, Suzhou, China. Association for Computational Linguistics.

Christina Heinze-Deml and Diane Bouchacourt. 2020. Think before you act: A simple baseline for compositional generalization. *CoRR*, abs/2009.13962.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does bert learn about the structure of language? In *ACL 2019-57th Annual Meeting of the Association for Computational Linguistics*.

Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 2017. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2901–2910.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, Dmitry Tsarkov, Xiao Wang, Marc van Zee, and Olivier Bousquet. 2020. Measuring compositional generalization: A comprehensive method on realistic data. In *International Conference on Learning Representations*.

Najoung Kim and Tal Linzen. 2020. COGS: A compositional generalization challenge based on semantic interpretation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9087–9105, Online. Association for Computational Linguistics.

Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. 2020. Attention is not only a weight: Analyzing transformers with vector norms. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7057–7075, Online. Association for Computational Linguistics.

Yen-Ling Kuo, Boris Katz, and Andrei Barbu. 2021. Compositional networks enable systematic generalization for grounded language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 216–226, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Brenden Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2873–2882. PMLR.

Brenden M Lake. 2019. Compositional generalization through meta sequence-to-sequence learning. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Yuanpeng Li, Liang Zhao, Jianyu Wang, and Joel Hestness. 2019. Compositional generalization for primitive substitutions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4293–4302, Hong Kong, China. Association for Computational Linguistics.

Chenyao Liu, Shengnan An, Zeqi Lin, Qian Liu, Bei Chen, Jian-Guang Lou, Lijie Wen, Nanning Zheng, and Dongmei Zhang. 2021. Learning algebraic recombination for compositional generalization. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1129–1144, Online. Association for Computational Linguistics.

Runtao Liu, Chenxi Liu, Yutong Bai, and Alan L Yuille. 2019. Clevr-ref+: Diagnosing visual reasoning with referring expressions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4185–4194.

Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *Advances in neural information processing systems*, 32.

Barbara Partee et al. 1984. Compositionality. *Varieties of formal semantics*, 3:281–311.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Arkil Patel, Satwik Bhattamishra, Phil Blunsom, and Navin Goyal. 2022. Revisiting the compositional generalization abilities of neural sequence models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 424–434, Dublin, Ireland. Association for Computational Linguistics.

Linlu Qiu, Hexiang Hu, Bowen Zhang, Peter Shaw, and Fei Sha. 2021. Systematic generalization on gSCAN: What is nearly solved and what is next? In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2180–2188, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. 2021. Do vision transformers see like convolutional neural networks? *Advances in Neural Information Processing Systems*, 34:12116–12128.

Laura Ruis, Jacob Andreas, Marco Baroni, Diane Bouchacourt, and Brenden M Lake. 2020. A benchmark for systematic generalization in grounded language understanding. *Advances in neural information processing systems*, 33:19861–19872.

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

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models.

Gail Weiss, Yoav Goldberg, and Eran Yahav. 2021. Thinking like transformers. In *International Conference on Machine Learning*, pages 11080–11090. PMLR.

Zhengxuan Wu, Elisa Kreiss, Desmond Ong, and Christopher Potts. 2021. ReaSCAN: Compositional reasoning in language grounding. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*.

A Model Architecture

The model architecture of our proposed approach GroCoT is shown in Figure 5.

B Implementation Details

We use PyTorch (Paszke et al., 2019) for all our implementations. All our models were trained from scratch, and the parameters were updated using Adam optimizer. We designed a compositional validation set by taking few examples from each compositional splits of the respective dataset. The best model is selected based on the accuracy on this compositional validation set. Hyperparameter tuning was done using grid search. We show the best hyperparameters for our models corresponding to different datasets in Table 10. Moreover, we show the average performance of 3 different runs with random seeds for all the models in the paper. We used 8 NVIDIA Tesla V100 GPUs each with 32 GB memory for all our experiments.

C Details of Datasets

C.1 ReaSCAN

ReaSCAN consists of around 500K train, 30K validation, and 6K test examples where each example is a pair of command and world state. Given these two as input, models are supposed to output the correct sequence of action tokens. Apart from the above splits, ReaSCAN also has 7 systematic generalization test splits in total. ReaSCAN has three types of input commands.

- `Simple:=` `$VV` `$ADV?` (equivalent to gSCAN commands)
- `1-relative-clause:=` `$VV` `$OBJ` that is `$REL_CLAUSE` `$ADV?`
- `2-relative-clauses:=` `$VV` `$OBJ` that is `$REL_CLAUSE` and `$REL_CLAUSE` `$ADV?`

See Table 11 for expansions of non-terminals in the grammar. Below, we describe the ReaSCAN test splits in detail:

A1 *Novel Color Modifier*: For this split, the train set never contains "yellow square" in the command. However, commands containing expressions such as "yellow circle" and "blue square" are present in the training set. During test time, this split expects model to zero-shot generalize on the phrase "yellow square".

A2 *Novel Color Attribute*: Here, the examples in which red squares are targets are held out from the train set. Commands in the train set also never contain the phrase "red square". However, the train set may contain red square objects as distractors in the background. This particular split tests model performance on novel combination of target object's visual attributes.

A3 *Novel Size Modifier*: In this split, a particular combination of size and shape attributes is held out from the train set. Specifically, the model never sees phrases like "small cylinder" or "small green cylinder" during training. While testing, the model needs to generalize to examples where cylinders of any color are referred using the "small" size attribute. Additionally, size being a relative concept in ReaSCAN, adds another level of complexity. For instance, size "small" can refer to an object of size 2 in a particular example, and in another example size "small" can instead refer to an object of size 3, depending on the other objects present in the grid world.

B1 *Novel Co-occurrence of Objects*: In this split, the commands contain those objects which never co-occur in training (e.g. "small red circle" and "big blue square"). However, models do encounter these objects co-occuring with other objects during training. In summary, this split tests whether models can generalize over novel co-occurrence of objects.

B2 *Novel Co-occurrence of Relations*: Commands containing both "same size as" and "inside of" relations are held out from the training data. At test time, models must generalize to the novel co-occurrence of these two relations. Importantly, during training model does encounter commands where the relation "inside of" co-occurs with other relations except "same size as".

C1 *Novel Conjunctive Clause Length*: This split contains commands with additional conjunctive clause i.e. the commands contains 3 relative clauses (e.g. "push the small red circle that is in the same column as a big green square and inside of a big blue box and in the same row as a blue square hesitantly"). Models trained with up to 2-relative clauses must generalize to longer commands which contain 3 relative clauses.

C2 *Novel Relative Clauses*: In ReaSCAN train data, commands contain a maximum of 1 recursive relative clause (e.g. "push the circle that is in the same column as a yellow square and inside of a

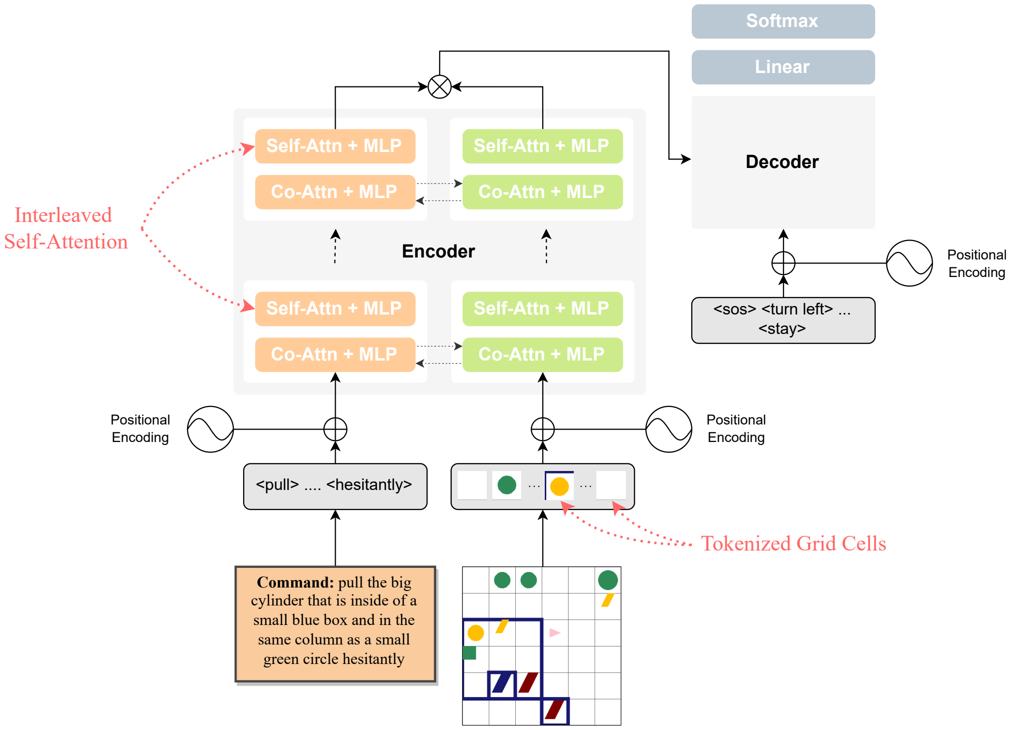


Figure 5: Illustration of the architecture for our proposed GroCoT model.

Hyperparameters	**ReaSCAN**	**GSRR**	**gSCAN**
# Self-attention Layers (Vision)	6	3	3
# Self-attention Layers (Language)	6	3	3
# Co-attention Layers	6	3	3
# Decoder Layers	6	6	6
Embedding Size	128	128	128
Hidden/FFN Size	256	256	256
Attention Heads	8	8	8
Learning Rate	[0.00005, 0.00008, **0.0001**]	[0.00005, 0.00008, **0.0001**]	[**0.00005**, 0.00008, 0.0001]
Batch Size	[**32**, 64]	[32, **64**]	[32, **64**]
Dropout	0.1	0.1	0.1
# Parameters	4.5M	3M	3M
Epochs	100	100	100
Avg Time (Overall)	64	14	18

Table 10: Different hyperparameters considered for models trained on ReaSCAN, GSRR, and gSCAN. Best hyperparameters for each model are in bold. (Avg Time shown above is in hours).

big box cautiously"). However, this test split contains commands with 2-recursive relative clauses i.e. there are two "that is" clauses in the commands (e.g. "push the blue circle that is in the same column as a blue cylinder that is in the same row as a green square hesitantly").

C.2 gSCAN

gSCAN is very similar to ReaSCAN; both are essentially grounded navigation tasks. The grid world in gSCAN is the same as that of ReaSCAN, although commands in ReaSCAN are much more complicated than gSCAN. Commands in ReaSCAN contain relative clauses, whereas gSCAN commands have no relative clauses. For example command from gSCAN looks like *walk to a red big square*. gSCAN contains around 350K training and 20K test examples for the compositional splits. Below, we briefly describe the individual compositional test splits in gSCAN:

A *Random*: This test split contains random examples and is supposed to test in-distribution generalization.

B *Yellow squares*: For this split, the train set doesn't contain examples where *yellow squares* are referred with color and shape attributes.

C *Red Squares*: Here, the examples where the target is *red square* are held out from the train set.

D *Novel Direction*: For this split, the examples where the target objects are located south-west of

Syntax	Descriptions	Expressions
`$VV`	verb	{walk to, push, pull}
`$ADV`	adverb	{while zigzagging, while spinning, cautiously, hesitantly}
`$SIZE`	attribute	{small, big}*
`$COLOR`	attribute	{red, green, blue, yellow}
`$SHAPE`	attribute	{circle, square, cylinder, box, object}
`$OBJ`	objects	(a | the) `$SIZE? $COLOR? $SHAPE`
`$REL`	relations	{in the same row as, in the same column as, in the same color as, in the same shape as, in the same size as, inside of}
`$REL_CLAUSE`	clause	`$REL $OBJ`

Table 11: Definitions of syntax used in ReaSCAN command generation.*the actual size of any shape is chosen from {1,2,3,4} as in gSCAN (Ruis et al., 2020). This table is taken from Wu et al. (2021).

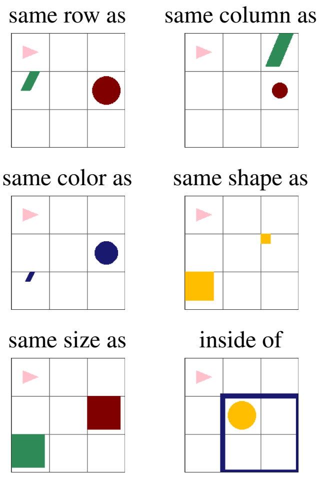


Figure 6: Relations.

the agent, are held out from training set.

E *Relativity*: To create this split, examples where circles with size 2 are referred as *small* are held out from the train set.

F *Class Inference*: In this split, all examples where the verb is *push*, and target is a square of size 3 are held out from the training set. Note that the model needs to infer that this object is of class 'heavy' based on the size 3 and needs to push twice to move an object by one grid cell.

G *Adverb k=1*: Only one example with the adverb *cautiously* is shown in the train set.

H *Adverb to Verb*: For this split, examples where the commands have verb *pull* and adverb *while spinning* are held out from the train set.

C.3 GSRR

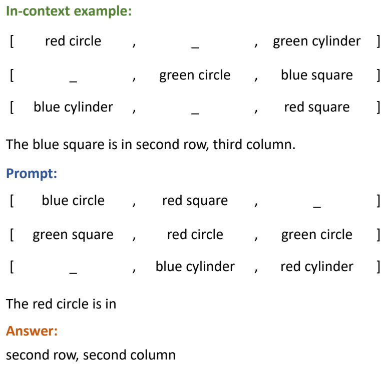
Based on the gSCAN task setup, (Qiu et al., 2021) proposed 5 additional compositional generalization splits that also test spatial reasoning capabilities of models. GSRR contains around 250K train examples, from which specific kind of examples are held-out to create the systematic splits. Below, we describe the compositional splits in GSRR:

I *Random*: This split contains random examples for testing in-distribution generalization.

II *Visual*: For this split, the train set doesn't contain examples where *red squares* are present either as targets or referent object.

III *Relation*: Here, the examples which contain both *green squares* and *blue circles* are held-out from the training data.

IV *Referent*: For this split, the examples where *yellow squares* are referred as target are held-out from the trainset.



In-context example:

[red circle , _ , green cylinder]

[_ , green circle , blue square]

[blue cylinder , _ , red square]

The blue square is in second row, third column.

Prompt:

[blue circle , red square , _]

[green square , red circle , green circle]

[_ , blue cylinder , red cylinder]

The red circle is in

Answer:

second row, second column

Figure 7: An example from the `direct-grounded` experiment on LLMs.

V *Relative Position 1*: Here, all examples where the targets are north of their referent objects are held-out from the training set.

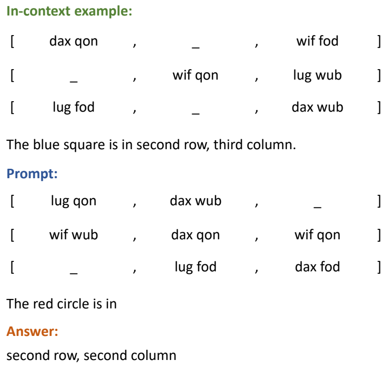
VI *Relative Position 2*: In this split, all examples where the target is south-west of the referent object are not seen in training data.

D Performance of Large Language Models

We design a simpler version of the task to evaluate the performance of large language models (LLMs) such as GPT-3 (Brown et al., 2020) and Codex (Chen et al., 2021). Given a 3 × 3 grid world and a simple command stating an object's color and shape, the model needs to output that object's location. The world state as well as the output are in a textual description format to make the task compatible with the input and output space of LLMs. Along with each test example, the models are given

Model	A	Comp. Average	B	C	D	E	F	G	H
Multimodal LSTM (Wu et al., 2021)	97.7	32.7	54.9	23.5	0.0	35.0	92.5	0.0	22.7
GCN-LSTM (Gao et al., 2020)	98.6	-	99.1	80.3	**0.2**	87.3	99.3	-	**33.6**
Multimodal Transformer (Qiu et al., 2021)	**99.9**	60.0	99.9	99.3	0.0	99.0	99.9	0.0	22.2
GroCoT (ours)	**99.9**	**60.4**	**99.9**	**99.9**	0.0	**99.8**	**99.9**	0.0	22.9

Table 12: Performance of GroCoT on gSCAN (Ruis et al., 2020).



In-context example:

[dax qon , _ , wif fod]

[_ , wif qon , lug wub]

[lug fod , _ , dax wub]

The blue square is in second row, third column.

Prompt:

[lug qon , dax wub , _]

[wif wub , dax qon , wif qon]

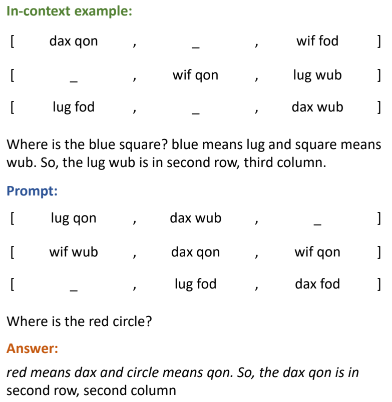
[_ , lug fod , dax fod]

The red circle is in

Answer:

second row, second column

Figure 8: An example from the `nonsense-grounded` experiment on LLMs.



In-context example:

[dax qon , _ , wif fod]

[_ , wif qon , lug wub]

[lug fod , _ , dax wub]

Where is the blue square? blue means lug and square means wub. So, the lug wub is in second row, third column.

Prompt:

[lug qon , dax wub , _]

[wif wub , dax qon , wif qon]

[_ , lug fod , dax fod]

Where is the red circle?

Answer:

red means dax and circle means qon. So, the dax qon is in second row, second column

Figure 9: An example from the `chain-of-thought` experiment on LLMs.

prompts containing multiple in-context examples. Note that the context provided to the model is ensured to contain all necessary information that the model might need to answer the test example.

In the most basic version of the experiment, called `direct-grounded`, we directly refer to the objects in the grid world with their attributes used in the command. See Fig. 7 for an illustration. We provide 30 in-context train examples to the model as part of the prompt for each of the 20 test example. In this setup, Codex achieved 95% accuracy while GPT-3 achieved 65% accuracy. This experiment merely serves as a sanity-checking baseline for the other experiments we described next.

Ideally, the model should learn the mappings of words in the commands to the tokens in the world state. Hence, in this experiment, called `nonsense-grounded`, we use non-sense words to refer to the objects in grid world as shown in Fig. 8. This task setting more closely resembles the target identification task in ReaSCAN (while being a much more simpler version of it). In this setup, the models fail badly. Codex achieves only 25% accuracy, and GPT-3 achieves only 30%. This clearly shows that LLMs are as yet unable to tackle such grounded compositionality tests, even when provided with sufficient evidence via in-context training examples in the prompt.

Following recent work (Wei et al., 2022), we provide explicit *chain-of-thoughts* to the LLMs to make them understand the task. While from a purely evaluation point-of-view, this can be considered *cheating*, we were merely curious to check whether the chain-of-thought idea, which has led to so much success in reasoning tasks, would help the models do better in this task setting. An illustration of the chain-of-thought provided to the models is given in Fig. 9. Codex performs much better when provided with such chain-of-thoughts, achieving 70% accuracy. However, GPT-3 still struggles on the task, achieving only 25% accuracy.

E Additional Details and Results of Analysis Experiments

E.1 Target Identification vs Navigation: What is the Challenge?

E.1.1 Comparison with Other Methods

Studies with similar objectives to ours have been carried out by Ruis et al. (2020) and Qiu et al. (2021). However, the results of their experiments are not very conclusive. Ruis et al. (2020) examined the object in the grid world that was being maximally attended by the agent and checked whether it is the target. However, their results do not correlate well with the conclusions. For instance, they find that in the error cases on the gSCAN 'C' split, the agent attends over the correct target object half the time! Qiu et al. (2021) compare the final position

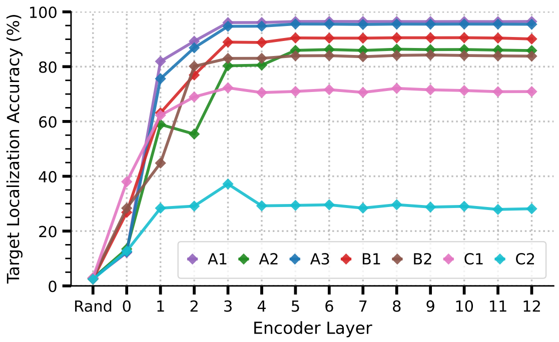


Figure 10: Probing target location information in encoder representations at different layers. Layer 0 corresponds to directly training a linear layer over the input embeddings and Rand corresponds to training a linear layer on random embeddings.

of the agent with the ground-truth target location. However, such an analysis fails to disentangle the subtasks of target identification and agent navigation and discards the causal relationship between them. Also, because of the interpretation of verbs such as *push* and *pull*, the final position of the agent may be very different from the target location.

E.1.2 Predicting Target Location from Earlier Encoder Representations

We experimented with predicting the target location from earlier encoder representations, including the embeddings and randomly initialized representations. The latter two act as baselines for our probing results described in section 4.1. The results are provided in Figure 10.

E.1.3 Target Identification Results on gSCAN

We show that target identification is the main challenge for most of the gSCAN splits as well. Similar to the experiment described in section 4.1, we experiment with providing ground-truth target locations to the model. As seen from the results provided in Table 13, identifying the target location is the main challenge for most of the splits in gSCAN.

E.2 Issues in ReaSCAN Test Set Design

E.2.1 Intuitive Explanation of Unfairness in C2 Split Design

In this section, we explain why the model would not be able to disambiguate between *'and'* and *'that is'* based on the train set. Since the meanings of *'and'* and *'that is'* are apparent to humans, to understand things from the model's perspective, let us replace them with non-sense words *'axyo'* and *'tafyo'* respectively. The model sees commands such as "walk to the small red square **tafyo** in the same row as the big blue cylinder" and "walk to the small red square **tafyo** in the same row as the big blue cylinder **axyo** in the same column as a green circle" during training. There isn't enough information here to make the model understand that *'tafyo'* applies the constraint on its right to the clause on its immediate left while *'axyo'* applies the constraint on its right to the clause that occurs on the immediate left of the *'tafyo'* before it. Looking at these examples, both *'tafyo'* and *'axyo'* do the exact same thing, i.e., apply the constraint on their immediate right over the first clause in the command.

SPLIT	ACCURACY
B	99.9
C	100
D	0.0
E	100
F	99.9
G	0.0
H	23.4

Table 13: Results of GroCoT on gSCAN when provided with ground-truth target locations in the input.

Splits	Action Seq Accuracy (Default)	Action Seq Accuracy ("and" replaced)	Consistency
A1	99.40	99.26	99.61
A2	88.74	88.41	96.86
A3	98.13	97.54	98.90
B1	93.55	94.43	97.36
B2	86.15	87.19	95.33
C1	74.06	68.90	72.78

Table 14: Measuring the consistency of the predictions made by the model before and after replacing all *and*'s with *that is*'s in ReaSCAN.

E.2.2 Experimental Details of Evaluation on C2-deeper

We randomly select 100,000 examples from the train set and 6,000 examples from the C2 test set. This forms the new train set. We generate 4500 new examples of depth three to form the C2-`deeper` test set.

F Additional Details and Results on our RefEx Task

F.1 Differences with Other Benchmarks

RefEx dataset is different from existing similar-looking synthetic benchmarks like SHAPES (Andreas et al., 2016), CLEVR (Johnson et al., 2017), and CLEVR-Ref+ (Liu et al., 2019). RefEx aims to test the systematic generalization capabilities of neural models in a grounded setting while keeping the task simple enough to allow easier interpretation of the model's behaviour. On the contrary, previous diagnostic benchmarks are more concerned with testing the overall reasoning capabilities of the model. Also, the close resemblance between RefEx and ReaSCAN allows us to use insights gained from the RefEx task to improve performance on ReaSCAN.

F.2 Details about the Task and Model

F.2.1 Details About the Task

Each variant in RefEx contains 90K training, 2.5K validation, and 2.5K test examples.

A1 We hold out all examples where the command contains "green square". As a result, the model never sees a *green square* object as the target although green squares occur in the background in the train set. This split expects the model to zero-shot generalize over the composition of "green" and "square" attributes.

A2 We hold out all examples where the command contains "red circle" while ensuring that model never encounters *red circle* object during training.

A3[12] We hold out all examples where the command is "small green circle" and the corresponding target is of size 2, meaning that the model has never seen a green circle of size 2 being referred to with "small".

A4[12] We hold out all examples where the command is "small blue cylinder". At test time, the model needs to zero-shot compose the concept of "small" with "blue cylinder" objects.

F.2.2 Details About the Model

Below, we describe the details of our attention-only transformer. We begin by mapping the command and the world state to d_{model} size embeddings using our embedding matrix. These embeddings constitute the initial input $\mathbf{X} \in \mathbb{R}^{d_{\text{model}} \times n}$ to our network, where n is the sequence length. Here, $n = 2 + 36$ for `two-attr` variant, $n = 3 + 36$ for `three-attr` variant, and $n = 8 + 36$ for `three-attr-rel` variant. Here, 2, 3, and 8 correspond to the tokens in the referring expression. Note that this input representation $\mathbf{X}$ contains information from both modalities. This representation is fed into the first multi-head attention block, and the subsequent outputs are residually added back to the initial input. We repeat this mechanism for successive attention blocks, and after the n^{th} attention block, the final representations corresponding to the world state tokens are mapped to logits by taking element-wise sum along the d_{model} dimension. In the end, we apply softmax operation on the logits for 36-way classification, where each class corresponds to a particular grid cell. The architecture is illustrated in Figure 11.

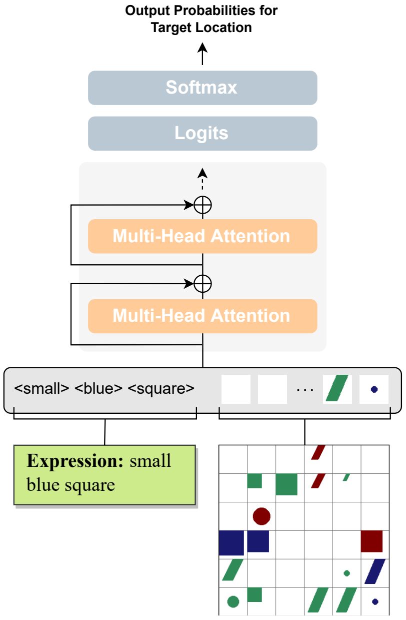


Figure 11: Model architecture used for the RefEx experiments.

The only learnable parameters for our model come from the query, key, value, and output matrices of different attention layers. For `two-attr` and `three-attr` variants, we don't require positional information for the command while we incorporate learned positional embeddings for the `three-attr-rel` variant.

F.3 Understanding Model Performance on the RefEx A1 Split

When we first created the Referring Expressions train and test sets, the attention-only transformer achieved perfect generalization on all splits except A1. On A1, the model achieved average accuracy.

[12] Only for the `three-attr` variants.

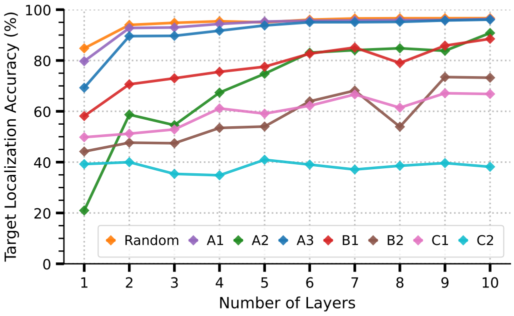


Figure 12: Target Localization accuracy for attention-only transformer on ReaSCAN. The model's performance plateaus after 3 layers for most of the compositional splits.

This was very surprising because the model was able to solve A2, which seems a strictly harder task than A1. The only difference between the two splits is that in A1, the held-out target object (*green square*) is seen as a distractor while in A2, the object (*red circle*) is never seen as a distractor. One probable hypothesis we had for the model failing on A1 was that the model was overfitting on the fact that *green squares* are always distractors in the train set, thereby preventing the model to predict it as a target at test time. To confirm this, we decrease the average number of *green square* distractors per example by about 75% in the train set and retrain the model. The model performance immediately jumps to 100%, thereby confirming our hypothesis.

Similarly, ReaSCAN also has a test split (A2), which is analogous to the A1 split in Referring Expressions. We observe that the Transformer model performs comparatively worse on A2 than other similar splits on ReaSCAN. Considering the above-mentioned result, we believe that the ReaSCAN A2 split is also suffers from the same issue. Hence, we modify the training distribution to vary the number of examples where "red square" object occurs as a distractor. We show our results in Figure 13. Note that the size of train set in this experiment is 200K examples which is less than half of the full ReaSCAN trainset. Even with less number of training samples, the model trained on the modified training distribution (No examples contain Red Squares as distractors) outperforms the model trained on full ReaSCAN, on the A2 split. This validates our hypothesis about the overfitting issue in transformers.

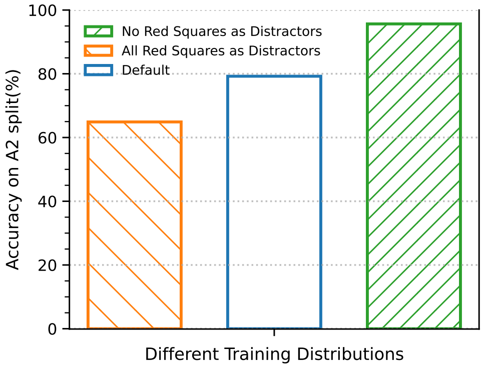
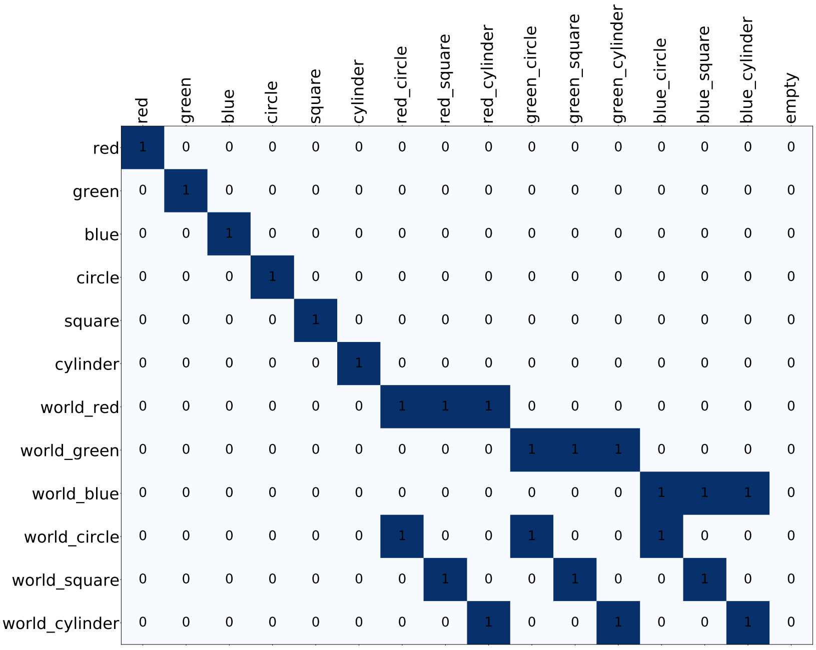
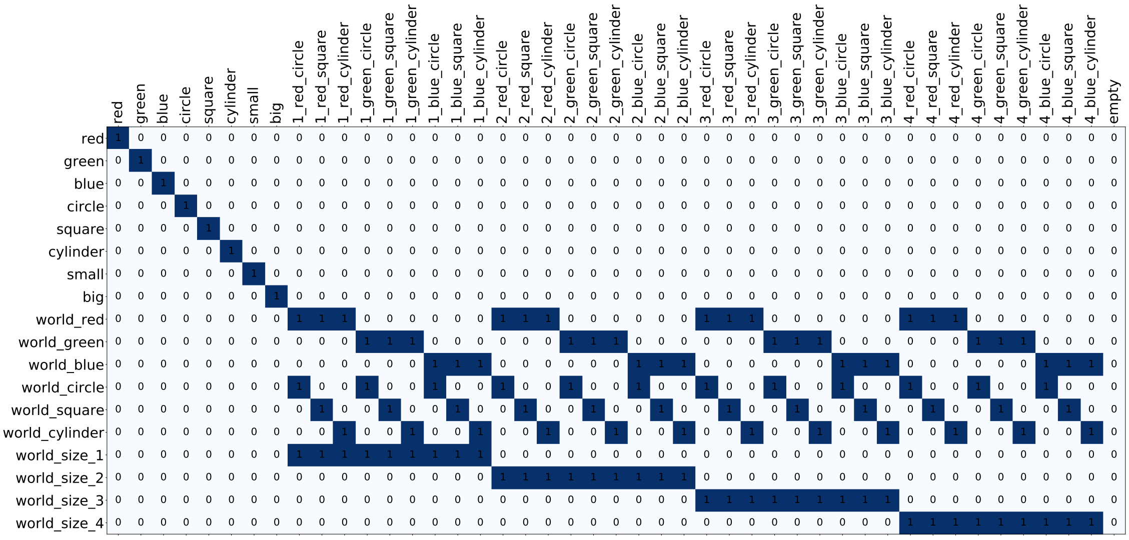


Figure 13: Effect of modifying training distribution on A2 accuracy. The blue bar corresponds to the original training distribution, the orange bar corresponds to a distribution where all red squares occur as distractors, and the green bar corresponds to the training distribution where no red square occurs as a distractor.



	red	green	blue	circle	square	cylinder	red_circle	red_square	red_cylinder	green_circle	green_square	green_cylinder	blue_circle	blue_square	blue_cylinder	empty
red	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
green	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
blue	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
circle	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
square	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
cylinder	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
world_red	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0
world_green	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0
world_blue	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0
world_circle	0	0	0	0	0	0	1	0	0	1	0	0	1	0	0	0
world_square	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0	0
world_cylinder	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0

Figure 14: Embedding matrix for `two-attr` variant. The columns correspond to different tokens in the vocabulary and the rows correspond to what each position in the embedding represents. Along columns, labels like *blue*, and *square* correspond to the command tokens, labels like *red_circle* correspond to the grid world tokens, where the object has "red" color attribute, and "circle" shape attribute, and the label *empty* corresponds to the grid world token where there is no object in the grid cell. Along rows, labels like *green* corresponds to the "green" color attribute for command tokens, and labels like *world_green* corresponds to "green" color attribute for grid world tokens.



	red	green	blue	circle	square	cylinder	small	big	1_red_circle	1_red_square	1_red_cylinder	1_green_circle	1_green_square	1_green_cylinder	1_blue_circle	1_blue_square	1_blue_cylinder	2_red_circle	2_red_square	2_red_cylinder	2_green_circle	2_green_square	2_green_cylinder	2_blue_circle	2_blue_square	2_blue_cylinder	3_red_circle	3_red_square	3_red_cylinder	3_green_circle	3_green_square	3_green_cylinder	3_blue_circle	3_blue_square	3_blue_cylinder	4_red_circle	4_red_square	4_red_cylinder	4_green_circle	4_green_square	4_green_cylinder	4_blue_circle	4_blue_square	4_blue_cylinder	empty
red	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
green	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
blue	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
circle	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
square	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
cylinder	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
small	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
big	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
world_red	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0
world_green	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	1	1	1	0	0	0	0
world_blue	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	1	1	1	0
world_circle	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	0
world_square	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0
world_cylinder	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0
world_size_1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
world_size_2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
world_size_3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
world_size_4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0

Figure 15: Embedding matrix for `three-attr` variant. The columns correspond to different tokens in the vocabulary and the rows correspond to what each position in the embedding represents. Along columns, labels like *blue*, *square*, and *big* correspond to the command tokens, labels like *1_red_circle* correspond to the grid world tokens, where the object has "1" size attribute, "red" color attribute, and "circle" shape attribute, and the label *empty* corresponds to the grid world token where there is no object in the grid cell. Along rows, labels like *green*, *big* correspond to the "green" color attribute, and "big" size attribute respectively for command tokens, and labels like *world_green*, *world_size_2* corresponds to "green" color attribute, and "2" size attribute respectively for grid world tokens.

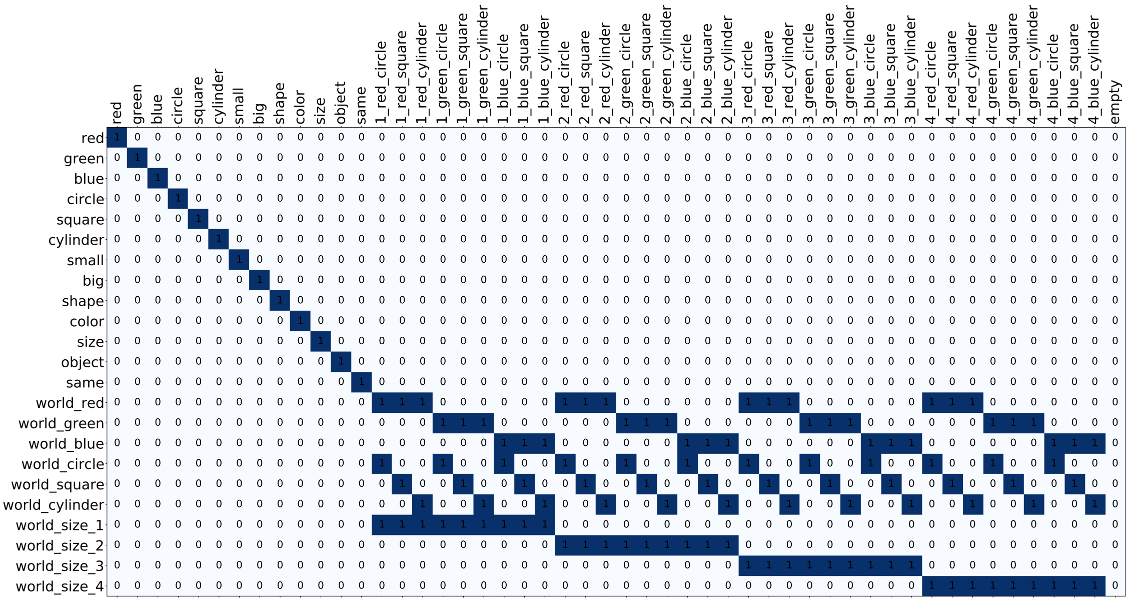
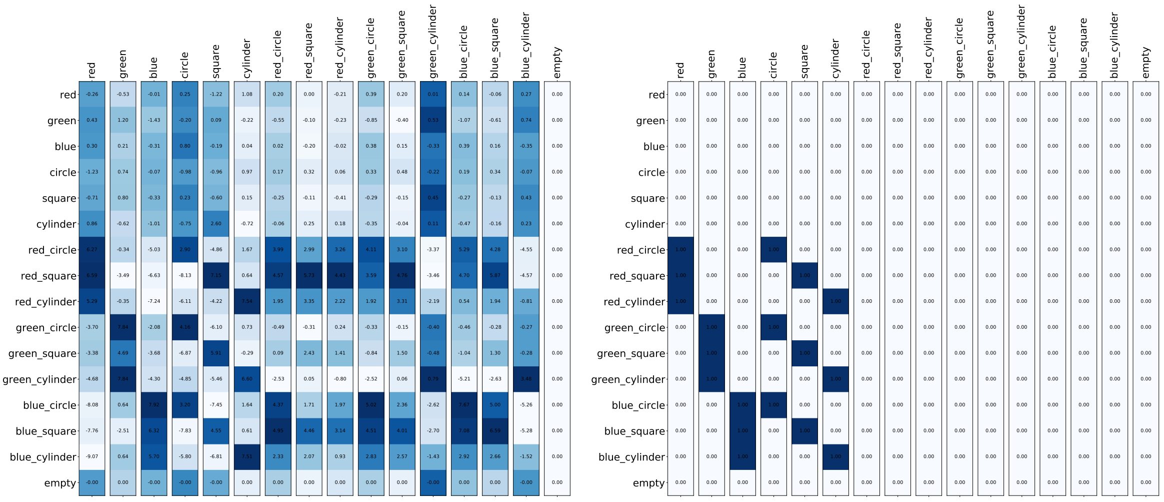


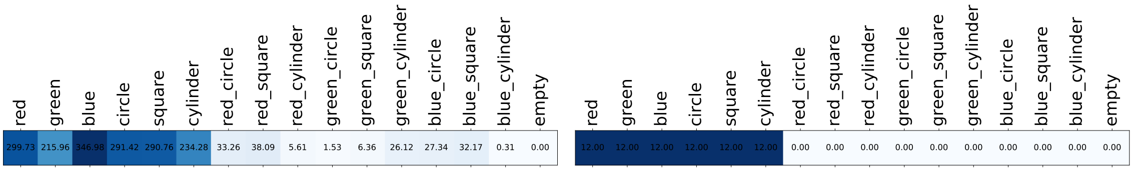
Figure 16: Embedding matrix for `three-attr-rel` variant. Rows and columns have the same meaning as described in Figure 15.



(a) Learned Model

(b) Our Construction

Figure 17: $\mathbf{M}_{\text{Learned}}$ (left side) for the attention-only transformer with a single layer and single attention head trained on the `two-attr` variant and our $\mathbf{M}_{\text{Construct}}$ matrix (right side) for the `two-attr` variant. Labels like *blue* and *square* correspond to the command tokens, labels like *red_circle* correspond to the grid world tokens, where the object has “red” color attribute and “circle” shape attribute, and the label *empty* corresponds to the grid world token where there is no object present in the grid cell. See section 5.3.1 for the exact formulation of $\mathbf{M}$ matrix.



(a) Learned Model

(b) Our Construction

Figure 18: $\mathbf{s}_{\text{Learned}}$ (left side) for the attention-only transformer with a single layer and single attention head trained on the `two-attr` variant and our $\mathbf{s}_{Construct}$ matrix (right side) for the `two-attr` variant. See section 5.3.1 for the exact formulation of $\mathbf{s}$. In this particular run, $\mathbf{s}_{\text{Learned}}$ contains positive scalars for command tokens. Some training runs also converged to negative values in $\mathbf{s}_{\text{Learned}}$ for command tokens. In that case, the interpretation of $\mathbf{M}_{\text{Learned}}$ changes accordingly, while the fundamental idea remains the same.

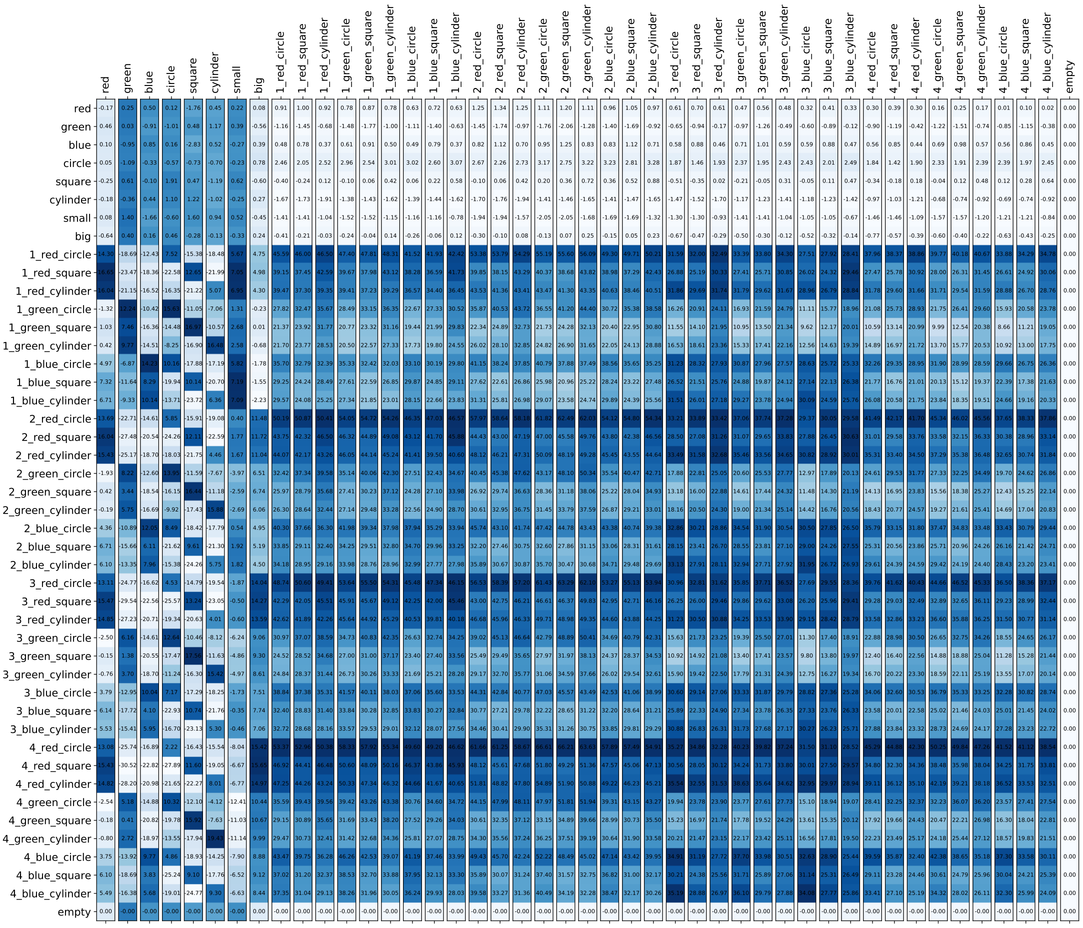


Figure 19: $\mathbf{M}_{\text{Learned}}$ for the attention-only transformer with a single layer and single attention head trained on the `three-attr` variant. Notice the matching pattern between $\mathbf{M}_{\text{Learned}}$ and $\mathbf{M}_{\text{Construct}}$ (shown in Figure 20). We can observe that the dot product between the key of an attribute and queries of grid world tokens with the corresponding attribute has higher values (darker grid cells). For example, in the column of "square" command token, the darker grid cells correspond to only those grid world objects which have the "square" shape attribute. Similarly, in the column for the "big" command token, grid world objects like *4_blue_square* with larger size have higher values as compared to objects like *1_blue_square* with smaller size.

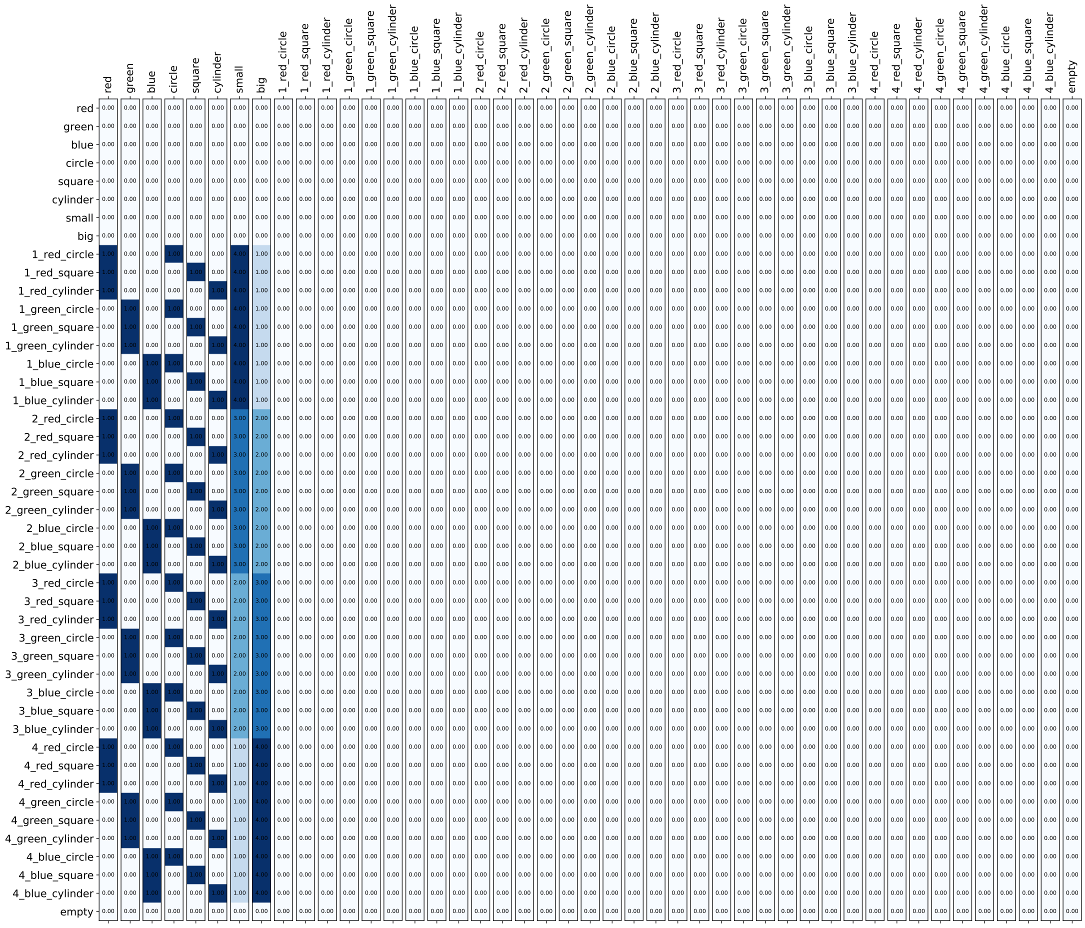


Figure 20: Our $\mathbf{M}_{\text{Construct}}$ matrix for `three-attr` variant.

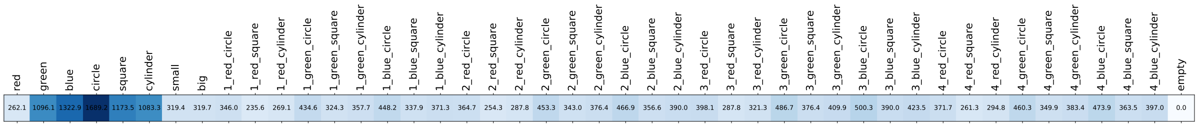


Figure 21: $\mathbf{s}_{\text{Learned}}$ for the attention-only transformer with a single layer and single attention head trained on the `three-attr` variant. In this particular run, $\mathbf{s}_{\text{Learned}}$ contains positive scalars for command tokens. Some training runs also converged to negative values in $\mathbf{s}_{\text{Learned}}$ for command tokens. In that case, the interpretation of $\mathbf{M}_{\text{Learned}}$ changes accordingly, while the fundamental idea remains the same.

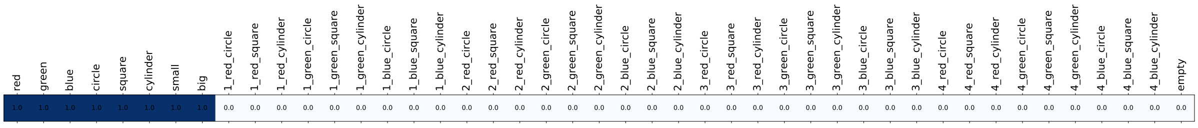


Figure 22: Our $\mathbf{s}_{\text{Construct}}$ for `three-attr` variant.