Mutual Exclusivity Training and Primitive Augmentation to Induce Compositionality

Yichen Jiang[*] **Xiang Zhou**[*] **Mohit Bansal**
UNC Chapel Hill
{yichenj, xzh, mbansal}@cs.unc.edu

Abstract

Recent datasets expose the lack of the systematic generalization ability in standard sequence-to-sequence models. In this work, we analyze this behavior of seq2seq models and identify two contributing factors: a lack of mutual exclusivity bias (one target sequence can only be mapped to one source sequence), and the tendency to memorize whole examples rather than separating structures from contents. We propose two techniques to address these two issues respectively: *Mutual Exclusivity Training* that prevents the model from producing seen generations when facing novel examples via an unlikelihood-based loss, and *prim2primX data augmentation* that automatically diversifies the arguments of every syntactic function to prevent memorizing and provide a compositional inductive bias without exposing test-set data. Combining these two techniques, we show substantial empirical improvements using standard sequence-to-sequence models (LSTMs and Transformers) on two widely-used compositionality datasets: SCAN and COGS. Finally, we provide analysis characterizing the improvements as well as the remaining challenges, and provide detailed ablations of our method.[1]

1 Introduction

Human intelligence demonstrates *systematic compositionality*, the algebraic capacity to understand and produce a potentially infinite number of novel combinations of known components (Chomsky, 1957; Montague, 1970), in comprehending and generating natural language. However, a number of recent datasets, e.g., SCAN (Lake and Baroni, 2018), COGS (Kim and Linzen, 2020), etc., provide clear evidence of the lack of systematic compositionality in state-of-the-art neural networks. By analogy with what humans do in meaningful learning (Ausubel, 1963; Shi et al., 2022), methods to solve this problem can be categorized into *deductive learning* and *inductive learning* methods. In *deductive learning*, the learner is first introduced to a general rule and then followed by specific examples where the rule is applied (Thornbury, 1999), while in *inductive learning*, the learner is first presented with abundant examples so that the learner can automatically infer the general rule by itself. Under the context of compositional generalization, *deductive learning* approaches directly inject useful priors (Li et al., 2019; Russin et al., 2020; Liu et al., 2020, 2021) to the model; while *inductive learning* approaches utilize data augmentation(Andreas, 2020; Akyürek et al., 2021) to provide more accurate examples that facilitate the learning of compositionality. In this work, we focus on designing methods that do not require any specific architectural changes, so it is applicable to standard sequence-to-sequence (seq2seq) models, and propose both deductive and inductive learning paradigms.

Our deductive method is derived from an observation that when facing compositionally novel examples, models have a tendency to generate old patterns seen during training, hence leading to numerous mistakes. Despite being a major challenge for models, humans avoid these mistakes by exploiting the *mutual exclusivity (ME) bias* (Markman and Wachtel, 1988), i.e., if a concept is already associated with one expression, humans are less likely to associate a new expression to that concept. Consequently, when given an unseen test input (expression), humans have the prior that this new input should not be mapped to a seen output (concept) that is already mapped to another input expression. Inspired by the ME bias, we propose a novel training framework **M**utual **E**xclusivity **T**raining (MET). Given an input-output pair, MET encourages the model to assign a high probability to the output given the input, and to *not* generate this out-

[*]Equal contribution.
[1]Our code is available at https://github.com/owenzx/met-primaug

put given any other inputs. We achieve this by first creating a randomly perturbed version of the input, and then adopting the unlikelihood loss (Welleck et al., 2019) on the perturbed input to penalize the generation of the original output. Finally, we train the model using a joint loss of MLE on the original batch and the unlikelihood loss. Additionally, we explore another variant MET-Meta, where we embed the unlikelihood loss as the meta generalization objective of the MAML framework (visualized in Fig. 1). This meta-learning objective will provide more regularization on the gradient updates instead of the final optimization target. On COGS and SCAN, we see substantial improvements from both variants of MET as long as the baseline seq2seq model has acceptable performance. MET improves a Transformer model from 76.1% to 80.6% on COGS, and improves an LSTM model from 13.5% to 38.7% on SCAN MCD2. The two MET variants also show interesting behavior differences. For Transformers, MET shows advantages on COGS, and MET-Meta show advantages on most of the SCAN tasks.

Despite the above improvements, a Transformer model equipped with MET may still struggle with very poor baselines. We hypothesize that this phenomenon may partially be because that the syntactic function to be generalized has not been applied to a sufficient number of distinct arguments in the training set. For instance, in the SCAN *Jump*, the function "*around left*" is only paired with four different lexical arguments ("*walk, run, look, turn*"). This lack of argument diversity makes it easier for the model to *memorize* all these patterns, but harder to infer the syntactic rules with *deductive learning*. To alleviate this memorization problem, we propose **prim2primX**, a data augmentation method to promote generalization by automatically generating a large set of new lexical arguments for each syntactic function. As the number of distinct arguments for each function increases, memorizing each single example independently becomes more challenging, while the difficulty of understanding the compositional structure remains the same, which makes it a more encouraging behavior. Specifically, the prim2primX procedure first builds a lexicon using a dataset-agnostic, rule-based word alignment algorithm. This lexicon contains the mapping between the input and the output form of each argument (e.g., "*run*$\mapsto$`RUN`"), while ignoring functional words (e.g., "*around*") that only decide the syntactic structure of the outputs. Then, to enrich the set of lexicons, we mutate each primitive ("*run*$\mapsto$*run0*; `RUN`$\mapsto$`RUN0`"), and create new examples by swapping the primitive argument (prim) with its mutated form (primX) on both sides. Our experiments (see Sec. 4) provide evidence that supports our claim above: with 15 new primitives (5 per original primitive) added to the SCAN lexicon, the accuracy on the SCAN *Jump* test set reaches 63.65% from the original performance of 3.49%. On COGS, with 2 new primitives per original one, prim2primX also improves the Transformer performance from 76.14% to 80.07% accuracy. Furthermore, we show that combining prim2primX and MET can achieve further improvement. Our prim2primX+MET achieves 81.1% accuracy on COGS, and prim2primX+MET-Meta achieves 74.0% on SCAN *Jump*. Finally, we also provide detailed ablations about each component in our proposed methods, and an analysis characterizing our improvements and remaining challenges.

Overall, our contribution is two-fold: (1) we propose MET, a novel, deductive training framework to inject mutual exclusivity bias into models; and (2) we propose prim2primX, a data augmentation method that automatically creates new arguments to facilitate inductive compositionality learning. Both methods and their combination show substantial empirical improvements on SCAN and COGS. Moreover, our methods are data and model-agnostic, and do not leak any test examples, so the improvements reveal the compositional generalization potential of general seq2seq models.

2 Background

2.1 Compositional Generalization Challenge

Compositional generalization challenges test models with *unseen combinations of seen structures and seen contents*. While neural models perform well on in-distribution examples, they fail on most compositionality challenges with a distributional shift between training and test examples even when some surface statistics (e.g., word frequency) are similar. One example dataset used in this work is **SCAN** (Lake and Baroni, 2018), which includes paired natural language command and action sequence outputs. We can represent this task as applying *syntactic functions* to the arguments.

Definition 1. *A **syntactic function** f is a symbolic function which maps certain input patterns to corresponding output structures.*

Function f	Examples		
	Input	**Program**	**Output**
$\mathrm{Rev}(x_1, x_2) = x_2 + x_1$	walk left	$\mathrm{Rev}(\mathrm{walk}, \mathrm{left})$	TL WALK
$\mathrm{Around}(x_1, x_2) = (x_2 + x_1) * 4$	walk **around** left	$\mathrm{Around}(\mathrm{walk}, \mathrm{left})$	TL WALK TL WALK TL WALK TL WALK
$\mathrm{Twice}(x) = x * 2$	walk opposite left **twice**	$\mathrm{Twice}(\mathrm{Oppo}(\mathrm{walk}))$	TL TL WALK TL TL WALK

Table 1: Syntactic function examples in SCAN (Lake and Baroni, 2018). "+" means concatenation. "$x * 2$" means replicating x twice. A full table containing all the syntactic functions in SCAN is in the Appendix.

Table 1 lists some example syntactic functions and corresponding input-output pairs in SCAN. The main challenge in SCAN *Jump* is to apply functions like $\mathrm{Around}(x_1, \mathit{left})$ to a lexical argument "*jump*", while such combination does not appear in the training set. Another dataset we use in this work is **COGS** (Kim and Linzen, 2020), which requires parsing a diverse set of natural language sentences into their corresponding logical forms. COGS raises five different generalization challenges: syntactic functions with novel (1) primitives or (2) modified phrases; (3) deeper recursion; (4) alternative verb argument; and (5) novel identification of a verb. Challenges 1, 4, and 5 require generalizing to unseen lexicons, while challenges 2 and 3 require generalizing to unseen structures.

2.2 Limitations of Standard Seq2Seq Models

Standard seq2seq models often struggle facing compositional challenges and fall into a common mistake pattern. For example, given a test command "*jump left twice*" that requires generalizing a syntactic function to "*jump*", Transformer sometimes generates the correct output structure but mistakenly picks another familiar primitive (e.g., "walk") over "*jump*", yielding "`TL` **`WALK`** `TL` **`WALK`**". This suggests that during the training, Transformer is not separately mapping the function "x *left twice*" to the output structure "`TL` x `TL` x"; and the argument "*walk*" to the action "`WALK`". As a result, when facing a novel command (e.g., "*jump left twice*"), in 79.8% of test cases, the model does not generate "`JUMP`" but instead repeats a seen training-set output. Additionally, in more than 63% of the test examples, the model cannot even correctly interpret the functional structure, e.g., it generates the structure for "*jump around left and run*" when it is required to "*jump around left and run thrice*". In contrast, humans (1) have a natural prior to not map two different sentences to the same point in the output space (Markman and Wachtel, 1988); and (2) are able to correctly separate syntactic structures from the actual contents (Lake et al., 2019). In the next section, we will show how we relieve this problem.

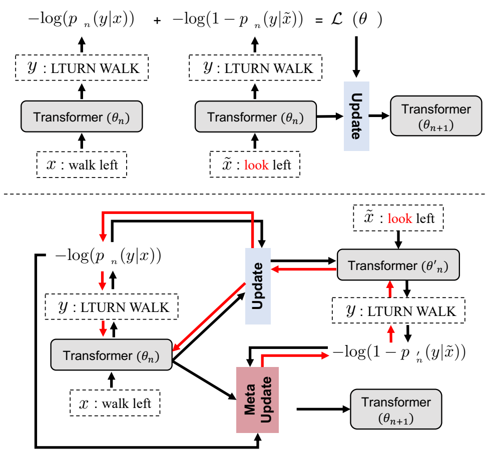
$-\log(p_{\theta_n}(y|x))$ + $-\log(1-p_{\theta_n}(y|\tilde{x}))$ = $\mathcal{L}(\theta)$
y : LTURN WALK
Transformer (θ_n)
x : walk left
$\tilde{x}$: look left
Update
Transformer (θ_{n+1})
Transformer (θ'_n)
$-\log(1-p_{\theta'_n}(y|\tilde{x}))$
Meta Update

Figure 1: **Upper**: MET. **Lower**: MET-Meta first updates the model with the likelihood loss, then performs a meta update with the sum of likelihood loss and meta-unlikelihood loss. The red arrows track the gradients generated by the meta-unlikelihood loss.

3 Methods

3.1 Injecting Mutual Exclusivity Bias

One crucial factor for humans to prevent the mistakes in Sec. 2.2 is the *mutual exclusivity bias* (Markman and Wachtel, 1988): if a concept is already associated with one expression, a human is less likely to associate a new expression to that concept. Under the context of seq2seq tasks, this bias should lead to the following behavior: in a bijective function task (i.e., the input and the output have one-to-one correspondence), when facing an unseen input, the model should not produce a seen output that is already associated with a seen input. Formally, we can represent it as:

Property 1. *Given a bijective training set D containing n input-output pairs (x_i, y_i). For a given test example $\hat{x}$, if $\forall i, \hat{x} \neq x_i$, then the label for the test example $\hat{y}$ will have the property $\forall i, \hat{y} \neq y_i$.*

Following this formalization, we propose a novel framework **M**utual **E**xclusivity **T**raining (MET) to inject this property into standard seq2seq models.

MET with Unlikelihood Loss. We notice that Property 1 can be achieved in a model-agnostic way by penalizing the predicted likelihood on $\hat{y}$, hence becoming a variant of *unlikelihood training* (Welleck et al., 2019). Intuitively, during training, for a given mini-batch (x, y), we sample some negative inputs $\tilde{x} \neq x$, which are similar to the original input x but **do not** map to the corresponding target y; and then we encourage the model to **not** generate y given $\tilde{x}$. Specifically, we train our model on the sum of maximum likelihood estimation (MLE) and unlikelihood loss:

$$\mathcal{L}_{\text{MLE}} = -\log(p(y|x)), \mathcal{L}_{\text{UL}} = -\log(1-p(y|\tilde{x}))$$

Notably, this unlikelihood loss $\mathcal{L}_{\text{UL}}$ is different from the loss used in Welleck et al. (2019), as the original unlikelihood is an average over all possible words, while the unlikelihood loss in Eq. 3.1 is operated at the sentence level. We adopt this change since sentence-level unlikelihood provides a suitable amount of regularization in our setting, while the regularization from the original word-level loss is too strong. See Appendix C for more discussions, proof-of-concept experiments and ablation results.

MET-Meta. MET encourages Property 1 in *the final trained model*. Alternatively, Property 1 can also be viewed as a regularization on the *learning behavior*, i.e., once a model is trained on a pair (x, y), it will give a lower probability to the pair with the same output but a different input $(\tilde{x}, y)$. We propose a variant: MET-Meta that achieves this effect by embedding the unlikelihood loss in a MAML (Finn et al., 2017; Conklin et al., 2021) framework. In MAML, each batch consists of two sets of examples, 'meta-train' and 'meta-test'. MAML encourages the model optimized on the meta-train examples to also optimize the loss on the meta-test, thus encouraging more generalizable behaviors. In MET-Meta, we use unlikelihood loss as the loss on meta-test. Specifically, with a sampled mini-batch (x, y) (meta-train), we first calculate the MLE loss and perform an SGD update $\theta'_n = \theta_n - \alpha\nabla_\theta\mathcal{L}_{\text{MLE}}(\theta_n)$, where α is the step size. Next, we use the unlikelihood loss on meta-test $(\tilde{x}, y)$ as the loss for the meta optimization w.r.t. the *updated* parameters θ'_n: $\mathcal{L}_{\text{UL}}(\theta'_n) = -\log(1 - p_{\theta'_n}(y|\tilde{x}))$. Finally, we optimize the sum of the MLE loss $\mathcal{L}_{\text{MLE}}(\theta_n)$ and the meta unlikelihood loss $\mathcal{L}_{\text{UL}}(\theta'_n)$ w.r.t. the *original* parameter θ_n to get the updated parameter θ_{n+1}:

$$\begin{aligned}\mathcal{L}_\tau(\theta_n) &= \mathcal{L}_{\text{MLE}}(\theta_n) + \mathcal{L}_{\text{UL}}(\theta'_n) \\ &= \mathcal{L}_{\text{MLE}}(\theta_n) + \mathcal{L}_{\text{UL}}(\theta_n - \alpha\nabla_\theta\mathcal{L}_{\text{MLE}}(\theta_n)) \\ \theta_{n+1} &= \theta_n - \beta\nabla_{\theta_n}\mathcal{L}_\tau(\theta_n)\end{aligned}$$

α and β are the step sizes for the two updates. A visualization of MET and MET-Meta is in Fig. 1.

Selecting the Best Negative Examples $\tilde{x}$. Similar to other training schema using multiple examples (e.g., contrastive learning (Gao et al., 2021), meta learning (Conklin et al., 2021), etc.), MET also depends on the quality of the negative examples $\tilde{x}$. One notable difference is that MET only uses input side $\tilde{x}$, and does not need the corresponding output $\tilde{y}$. This allows us to remove the constraint of only using seen training examples, and to use any good negative examples as long as it is similar to the positive example but semantically different. To achieve a similar goal, Conklin et al. (2021) mine examples from the training set via similarity metrics. However, the input distribution of these mined examples is constrained to the training set, and inherits the highly skewed training distribution. To overcome this limitation, we directly perturb the original training examples x to create $\tilde{x}$. Specifically, we randomly select one token in the training example, and replace the selected word with another word. In our preliminary experiments, we notice that it is important (1) to maintain the grammaticality of the perturbed sentence, as training on ungrammatical sentences can make the model easily overfit to the grammar errors; and (2) to have access to the whole vocabulary during training so that we can minimize the influence of a highly skewed training set distribution (a related discussion is at Sec. 7). Hence, we replace the selected words with other words sharing the same grammar properties. In this work, we do not delve in the direction of clustering the words with the same grammar properties and use simple approaches to test our ideas. We use an off-the-shelf tagger (Bird et al., 2009) to obtain the tags of words in COGS, and we cluster the words in SCAN together if they share the same immediate contexts. A similar effect can be obtained on more complicated datasets by several different approaches (Stratos, 2019; Akyürek and Andreas, 2022).

3.2 Diversifying Lexical Arguments with Primitive Augmentation

While our empirical results demonstrate the effectiveness of MET, we also notice that MET does not work on extremely weak baselines. In the training set of SCAN *Jump*, where a Transformer scores a poor 3.5% accuracy and does not improve with MET, each syntactic function (e.g., "*around left*") is only paired with four different lexical arguments ("*walk, run, look, turn*"). Such lack of argument variability in the dataset makes it very easy for the model to memorize all input-output mappings independently. We hypothesize that current datasets do not provide enough incentives for the model to infer the functional structures with *inductive learning*.Specifically, in order for a neural model to correctly learn a syntactic function f that can generalize to all the suitable arguments, the dataset must meet two hypotheses.

Hypothesis 1 (**Sufficient stimuli for function generalizability**). *f must be applied to a sufficient number of different arguments in the training set.*

Satisfying Hypothesis 1 ensures that models recognize f as a generalizable function instead of a special phrase (e.g., idioms) to memorize independently. To further generalize f to an unseen argument a, the dataset must suffice another hypothesis:

Hypothesis 2 (**Sufficient stimuli for argument equivalence**). *Suppose during training, function f is applied to a set of arguments B. Then in order to apply f to an unseen argument a, one of the following two conditions must hold:*

2.A *a and another argument $b \in B$ both appear in a sufficient number of distinct functions;*

2.B *a must appear in the same function with a sufficiently large subset $\bar{B} \subseteq B$.*

This hypothesis makes sure that there is enough stimuli to infer the syntactic equivalence between the new argument a and arguments in B, so that f can generalize to a. For example, in SCAN *Around Right*, primitives "*left*" and "*right*" both appear in a number of functions like "*walk left/right*", "*jump opposite left/right*", which meet Hypothesis 2.A above and help demonstrate the equivalence of "*left*" and "*right*". Therefore, to verify the effectiveness of these hypotheses, we propose the **prim2primX** data augmentation procedure to automatically enlarge the set of distinct lexical arguments that are applied to a syntactic function. and lead to empirical improvements in Sec. 4. We explain this procedure in the next few paragraphs (also see Alg. 1 in Appendix A).

Building a Lexicon. We first use a dataset-agnostic, rule-based word alignment algorithm to build a lexicon that maps a primitive's input form to its logical form (e.g., "*run* $\mapsto$ RUN"), while ignoring functional words (e.g., "*around*") that only decide the syntactic structure of the outputs. We denote the source vocabulary as V and the target vocabulary as W. The general intuition is that we want to find pairs $(v, w), v \in V, w \in W$ such that the presence of v in the input is both *necessary* and *sufficient* for the presence of w in the output.

$$\begin{aligned} \text{suff}(v, w) &= \forall (x, y), (v \in x) \rightarrow (w \in y) \\ \text{ness}(v, w) &= \forall (x, y), (w \in y) \rightarrow (v \in x) \end{aligned} \tag{1}$$

Enforcing these two conditions on the SCAN *Jump* task returns 6 primitive pairs, e.g., (*run*, RUN), (*walk*, WALK), etc. Successfully identifying them allows us to later swap them with new primitives to enlarge the set of distinct arguments applied to a function. However, for tasks with a larger vocabulary, a word's different forms are often parsed to the same output token. Hence we use the "*no-winner*" condition (Akyurek and Andreas, 2021) that only enforces $\text{suff}(v, w)$ and allows many-to-one mappings. This relaxation makes it possible to build a comprehensive lexicon on more complicated datasets (e.g., COGS) as it allows mappings like "walk $\mapsto$ WALK; walked $\mapsto$ WALK" that share the same target token. We refer to Appendix A.1 for details.

Mutating Primitives. With a primitive lexicon, we go over each example in the training set and randomly mutate some primitives (prim) by adding a suffix to their source and target forms (primX). Given an original example "*walk left twice*", we select a primitive "*walk*" and mutate it to get new examples like "***walk0*** *left twice* $\mapsto$ TL **WALK0** TL **WALK0**". This mutation operation can enhance Hypothesis 1 by creating more primitive arguments for each function. It also improves Hypothesis 2.B by creating a larger argument set B that is applied to the identity function. We argue that as the number of distinct arguments for each syntactic function increases, it becomes more challenging to memorize the corresponding output of each input, while the difficulty of compositionally separating function structures from argument symbols remains the same. Therefore, compositional generalization would become a more favorable solution to the model. Compared to previous data augmentations (Andreas, 2020; Akyürek et al., 2021) prim2primX has two advantages: (1) it avoids the difficult task of mining semantically equivalent input-output pairs that can be swapped as it brings in new primitives; (2) hence the augmented data will never reveal any test compositions, which leads to the model's generalization to future primitives.

Model	Overall	C1	C2	C3	C4	C5
MAML (Conklin et al., 2021)	64.1±3.2	-	-	-	-	-
Lex Learn* (Akyurek and Andreas, 2021)	82.0±0.0	-	-	-	-	-
Transformer Baseline	76.14±2.33	82.75±2.25	0.00±0.00	0.09±0.18	93.79±3.57	97.39±3.96
+ MAML	78.33±1.38	86.27±1.93	0.00±0.00	0.12±0.09	95.30±2.11	99.11±0.48
+ MET	**80.64**±1.08	**90.06**±2.09	0.00±0.00	0.23±0.46	**97.36**±0.97	**99.68**±0.36
+ MET-Meta	78.14±1.76	86.76±3.51	0.00±0.00	**0.73**±0.65	93.95±1.68	98.64±0.67
Transformer + prim2primX	80.07±1.22	88.39±2.48	0.00±0.00	0.77±0.64	97.59±0.78	**99.92**±0.08
+ MAML	80.93±0.38	89.86±0.70	0.00±0.00	**1.19**±1.31	**98.61**±0.35	99.20±1.05
+ MET	**81.12**±0.19	**90.25**±0.59	0.00±0.00	0.63±0.58	98.55±0.44	99.89±0.12
+ MET-Meta	80.96±0.43	90.15±0.88	0.00±0.00	0.52±0.43	98.24±0.62	99.75±0.44

Table 2: Test accuracy from the COGS (Kim and Linzen, 2020), including the overall result on the entire generalization test set and scores on each of the five challenges (C1, C2, C3, C4, C5 as discussed in Sec. 2.1). The model with * requires special architecture changes, hence is not directly comparable to our numbers. "+MAML" is our reimplementation of Conklin et al. (2021) on our own baselines.

4 Experiments

4.1 Experimental Setup and Baselines

We report results for both LSTMs and Transformers in our SCAN experiments. We split the original test set into a new development set that contains 10% of the examples and a new test set containing the rest 90%.[2] We select the checkpoint with the best dev-set accuracy. We refer to Appendix B.3 for more details of our experimental setup.

Additionally, in our experiment tables, we provide results for previous representative methods for a more complete picture of the comparison. We show results of the meta-learning method used in Conklin et al. (2021) as "+MAML" in the tables. To the best of our knowledge, this is one of the few baselines that do not require task-specific architecture changes to the model, similar to our method. Note that while both our method and Conklin et al. (2021) use the MAML framework (Finn et al., 2017), the training objectives are different: the meta-loss in Conklin et al. (2021) is the standard MLE loss, while ours is the unlikelihood loss. We discuss the combination of both ideas in Sec 4.4. Similar to the finding in Csordás et al. (2021), we notice different baseline configurations can have a substantial influence on results, hence we report the reimplemented MAML on our baselines using the Levenshtein distance for a fair comparison.[3] Additionally, for experiments on SCAN, we provide results from previous work with top task-specialized model architecture (Li et al., 2019; Liu et al., 2020), large pretrained model (Furrer et al., 2020), and state-of-the-art data augmentation (Andreas, 2020). On COGS, we provide results for Lex Learn (Akyurek and Andreas, 2021), which enhances a normal LSTM model with a special copying mechanism. Many of these baselines (e.g., CGPS, T5-11B, LANE, Lex Learn) require special architecture changes or pretraining models, so they are not directly comparable to our method.

[2]In all experiments, we report the mean accuracy in 5 runs and the standard deviation.

[3]Additionally, for SCAN MCD and COGS, we also provide the original results from Conklin et al. (2021). However, Conklin et al. (2021) do not report results on the SCAN *Jump* and *Around Right* splits, so we only compare to our reimplemented results on those two tasks.

4.2 COGS Experiment

Our results on COGS are shown in Table 2, and they also demonstrate the effectiveness of both MET and prim2primX. MET improves the Transformer performance from 76.14% to 80.64%, outperforming MAML and MET-Meta; prim2primX can bring additional improvements by creating extra lexical arguments, and pushes the best performance to 81.12%. More result discussions and an example showing how prim2primX works on COGS are shown in Appendix D.

Next, to have a better understanding of the behavior of current models, we break down the model accuracy on COGS to five different challenges. As explained in Sec. 2.1, challenges 1, 4, and 5 focus on generalizing a learned syntactic function to an unseen lexical argument, while challenges 2 and 3 focus on generalizing to unseen phrases (structural arguments). We notice both MET and prim2primX improve the generalization accuracy on challenges 1, 4, 5, reaching over 90% on all three challenges, showing promising progress in lexical generalization. However, challenges 2 and 3 remain to be two formidable tasks, as all our models constantly fall below 1% accuracy in test examples with novel

Model	*Jump*	*Around Right*
CGPS* (Li et al., 2019)	98.8±1.4	83.2±13.2
LANE* (Liu et al., 2020)	100.0	100.0
T5-11B† (Furrer et al., 2020)	98.3	49.2
GECA (Andreas, 2020)	87±2	82±4
LSTM Baseline	0.58±0.29	10.33±3.20
+ MAML	**0.81**±0.11	13.18±4.11
+ MET	0.14±0.07	**26.77**±11.14
+ MET-Meta	0.21±0.23	21.29±7.35
Transformer Baseline	**3.49**±1.65	19.86±10.41
+ MAML	1.87±0.48	**41.21**±8.63
+ MET	1.85±1.79	32.41±17.82
+ MET-Meta	1.78±1.52	36.61±7.16
LSTM + prim2primX	3.34±1.90	97.67±1.21
+ MAML	4.51±1.98	**99.79**±0.13
+ MET	7.34±5.58	97.63±0.99
+ MET-Meta	**7.51**±10.62	94.09±5.13
Transformer + prim2primX	63.65±15.47	84.50±15.32
+ MAML	**76.40**±19.82	**99.95**±0.04
+ MET	62.48±9.14	62.44±24.36
+ MET-Meta	73.94±20.68	65.63±20.34

Table 3: Test accuracy from the SCAN *Jump* and *Around Right* tasks. Methods with * require special architecture changes and the model with † is pre-trained on large corpora. Hence they are not directly comparable to our models.

Model	*Jump*	*Dax*
Transformer	3.49±1.65	0.64 ±0.68
+ GECA (Andreas, 2020)	87±2	6.31±7.06
+ Recombine (Akyürek et al., 2021)	**88.0**±7	7.85±3.17
+ prim2primX	63.65±15.47	65.35±16.10
+ prim2primX + MET	62.48±9.14	**75.80**±13.82

Table 4: Test accuracy on generalizing to an unseen primitive "*dax*".

structural arguments or deeper function recursion. We discuss this limitation and future remedial approaches in Sec. 7.

4.3 SCAN Results

Generalizing functional compositions to new lexical arguments. We report results on SCAN *Jump* and *Around Right* in Table 3. Both the LSTM and Transformer baselines perform poorly on generalizing learned functions to unseen lexical arguments, reaching < 20% accuracy on *Around Right* and < 4% accuracy on *Jump*. We first focus on the 2nd and 3rd row-groups in Table 3. On the *Around Right* task, with moderate baselines, MET can improve both Transformer (from 19.86% to 32.41%) and LSTM baselines (from 10.33% to 26.77%). Compared to the other deductive MAML methods (Conklin et al., 2021), the MET is better for LSTMs, but worse on Transformers. However, with an extremely weak baseline (< 4% accuracy), none of the deductive methods successfully improves the model on SCAN *Jump*. Next, the bottom two row-groups present the results using prim2primX data augmentation. By adding in total 15 additional new lexical arguments (not from the test dataset), prim2primX substantially improves the results on both LSTM and Transformer baselines. Most notably, it improves the Transformer performance from 3.49% to 63.65% on SCAN *Jump*, and from 19.86 to 84.50 on *Around Right*. With stronger baselines on *Jump*, our deductive methods can bring additional improvement. MET-Meta improves the performance from 3.34% to 7.51% for LSTM, and from 63.65% to 73.94% for Transformers. On these two tasks, we also see MET-Meta often outperforms the standard MET.

Model	MCD1	MCD2	MCD3
T5-11B† (Furrer et al., 2020)	7.9	2.4	16.8
LANE* (Liu et al., 2020)	100.0	100.0	100.0
AuxSeq* (Jiang and Bansal, 2021)	99.9±0.2	90.1±6.5	98.2±3.2
MAML (Conklin et al., 2021)	47.6±2.3	35.2±3.9	11.4±3.0
LSTM Baseline	21.49±3.30	13.52±3.49	13.11±2.13
+ MAML	24.27±3.30	11.80±2.13	12.33±1.37
+ MET	**33.20**±1.20	**38.66**±2.29	14.77±1.84
+ MET-Meta	31.31±2.23	36.20±2.75	**15.58**±2.30

Table 5: Test accuracy from the SCAN MCD tasks.

Generalizing to future unseen primitives. Compared to previous data augmentation methods (Andreas, 2020; Akyürek et al., 2021), our prim2primX has one crucial advantage: as it replaces original words with new primitives, it only encourages the model to induce compositionality from data and does not leak any test compositions. Oppositely, previous methods learn rules to uncover possible test examples, while models still struggle to generalize beyond the augmented data. As a result, models trained with prim2primX will generalize to future unseen primitives better. To demonstrate this, we design a new task *Dax* where we add the same number of "*dax*$\mapsto$DAX" into a few augmented SCAN *Jump* training sets and evaluate on test examples like "*dax around left twice*". Table 4 show that prim2primX maintains similar improvements [4] on *Dax* as *Jump*, while models trained with GECA and Recombine perform poorly on *Dax*.

[4]For prim2primX, "*jump*" and "*dax*" are very similar as both primitives exist as stand-alone examples during training. However, due to the high performance variance across different runs on SCAN (also common in other works), MET shows different performance on *Dax* (76.8%) and *Jump* (62.48%). Nonetheless, the difference here does not influence our claims.

Model	*Around Right* (L)	COGS (T)
MET	25.33±11.16	**80.52**±1.13
MAML	11.77±4.87	78.13±1.33
MET + MAML	**35.85**±19.37	80.37±0.95

Table 6: Dev set performance change by combining unlikelihood with MAML. L=LSTM, T=Transformer.

Model	*Jump* (T)	*Jump* (L)	COGS (T)
baseline	3.19±1.67	0.70±0.28	75.90±2.34
+ 1 prim	21.66±15.32	0.70±0.23	78.82±1.01
+ 2 prim	77.23±16.64	0.86±0.27	**80.14**±1.10
+ 3 prim	**77.58**±15.78	0.81±0.25	78.73±2.36
+ 4 prim	55.92±8.22	1.17±0.30	79.80±1.15
+ 5 prim	62.80±15.61	**3.30**±1.99	79.48±2.08

Table 7: prim2primX ablation, reported as dev accuracy from SCAN *Jump* and COGS. '+ 2 prim' means 2 new primitives per original primitive (e.g., run0 and run1).

Generalizing to new structural arguments (MCD Tasks). The results on three SCAN MCD tasks are presented in Table 5. As opposed to the two tasks above that focus on generalizing functions over lexical arguments, the MCD challenges additionally require the model to generalize syntactic functions to *structural* arguments, which contain their own hierarchical structures. We use LSTM as our baseline because previous work (Conklin et al., 2021) has shown that it outperforms Transformer significantly on SCAN MCD tasks, hence is more suitable for our deductive learning improvements. On all three tasks, MET brings substantial improvements to the baseline, with over 12% of average improvement, and both MET variants show similar performances and outperform the MAML method (Conklin et al., 2021).These results suggest that MET can improve the generalization to not only new lexical arguments, but also new structural arguments. We do not observe any improvement by training the LSTM on our augmented data that only create new lexical arguments. We discuss this limitation and possible future direction in Sec. 7.

4.4 Ablation Studies

We investigate the combination of deductive methods (MET and MAML), and the effect of different numbers of new arguments. More ablations about the unlikelihood loss are in Appendix C.2.

Combination of MET and MAML. Both MET and the MAML loss used in Conklin et al. (2021) aim to improve the deduction learning ability of models. Since the two methods come from two different angles, in theory they should also be complementary to each other. However, in Table 6, we see a mixed trend by combining MET and MAML. For example, on COGS, using both unlikelihood and MAML provide no additional gain, while there does appear to be sizable improvements on SCAN *Around Right*, with around 10% improvement on MET and 24% over MAML. These mixed trends indicate that while two deductive methods can have complementary improvements in some situations, but there also exist common limitation that is unsolvable to both methods and can lead to a diminishing gain from combining them.

Model	Structural Error	Primitive Error
Transformer	73.80±21.21	22.99±19.66
+ prim2primX	35.82±17.36	0.52±0.67
+ prim2primX + MET-Meta	25.90±23.03	0.16±0.21

Table 8: Structural and primitive error of best Transformer models on SCAN *Jump*.

Data Augmentation Ablation. We study how the different number of new arguments in prim2primX improves *inductive learning* of a Transformer. In Table 7, there is an overall trend that the model's performance improves as the number of new primitives and total training data increases. However, the gain from using additional data also saturates after a certain amount. The Transformer baseline requires 2 new primitives per original one to converge to its best performance on SCAN *Jump* and COGS, while LSTM needs 5 new primitives to achieve a significant improvement on *Jump*. Hence, in our experiments, we use 2 new primitives for COGS, and 5 new primitives for SCAN, where LSTM reaches its best performance, and Transformers still have decent accuracy.

4.5 Qualitative Analysis

Finally, we present a qualitative analysis into how effective are MET and prim2primX in inducing compositionality in Transformer and reduce the errors discussed in Sec. 2.2. Specifically, we inspect the generated outputs of the multiple best-seed models on SCAN *Jump* test set in Table 8. We categorize model errors into two categories: primitive errors, and structural errors. In primitive errors, the models generate the correct structure, but only choose the wrong primitive (e.g., the model generates the output of "*walks left twice*" when the input is "*jump left twice*"). In structural errors, the models make more significant errors in the generated structure of the output. In Table 8, 22.99% of Transformer baseline's errors are primitive error.

However, Training the model with prim2primX can reduce the error rate to 0.52%, and further to 0.16% by also using MET-Meta. The improvements suggest that prim2primX and MET can effectively prevent the model from associating a familiar concept to a new expression. Our methods reduce the structural error rate to 25.9% but can not completely eliminate them. How to further reduce structural errors remains a future direction.

5 Related Work

Compositional Generalization. Early work has studied neural networks' ability in systematic behavior (Wong and Wang, 2007; Brakel and Frank, 2009) in language learning, compositional counting ability (Wiles, 1998; Weiss et al., 2018) and syntax learning ability (Linzen et al., 2016). Many recent compositional generalization datasets (Lake and Baroni, 2018; Kim and Linzen, 2020; Loula et al., 2018; Liška et al., 2018; Bastings et al., 2018; Keysers et al., 2020; Tsarkov et al., 2021) greatly facilitate research in this direction. Previous research explored different ways to tackle the compositionality challenge, including grammar-based approaches (Nye et al., 2020), separating syntax and semantics (Li et al., 2019; Russin et al., 2020), architecture improvements (Dessì and Baroni, 2019; Gordon et al., 2020; Oren et al., 2020; Zheng and Lapata, 2021; Herzig et al., 2021; Shaw et al., 2021), task decomposition (Liu et al., 2020, 2021), semi-supervised learning (Guo et al., 2021), multi-task learning (Jiang and Bansal, 2021), meta-learning (Lake, 2019; Conklin et al., 2021), etc.

Data Augmentation. Most data augmentation (Andreas, 2020; Akyürek et al., 2021) methods promote generalization by creating extra data that resemble the test-set distribution, hence simplifying the problem. Qiu et al. (2021) sample new combinations of structures from a context-free grammar induced from the training data. Contemporarily, Akyürek and Andreas (2022) adopt a similar procedure to ours. They detect a primitive lexicon and then swap an original primitive with another one of the same types. In comparison, prim2primX focuses on introducing new primitives, and hence not exposing any novel test-set compositions. Also concurrently[5], Patel et al. (2022) create new primitives (e.g., "swim", "clap") and replace the original primitives (e.g., "walk") with these new ones to augment the training data. Similar to our findings, they reveal that the number of different lexical arguments greatly affects the model's generalization. They further show the limit of models under MLE training on SCAN *Jump* using larger models and much more augmented primitives. We instead focus on proving the effectiveness of inductive and deductive methods on widely-used baseline configurations and multiple datasets.

Mutual Exclusivity Bias. Mutual exclusivity (ME) bias helps children acquire the meaning of new words (Markman and Wachtel, 1988). However, vanilla neural networks do not possess this property (Gandhi and Lake, 2020). Recently, a number of works explore to inject ME bias into the models, mostly focusing on cross-situation word learning (Kádár et al., 2015; Lazaridou et al., 2016; Gulordava et al., 2020). On a high level, our work is most similar to Gulordava et al. (2020), where they use a margin-based loss for feed-forward networks in a word learning experiment. In comparison, our work uses an unlikelihood-based method for seq2seq generation problems. Some compositional generalization methods are also connected to ME biases (Lake, 2019; Akyurek and Andreas, 2021), but their specific architecture may not be generalizable to standard seq2seq models.

6 Conclusion

We propose two methods to improve compositional generalization: (1) MET, a mutual exclusivity bias-inspired method implemented with unlikelihood training; and (2) prim2primX, a data augmentation procedure automatically diversifying the arguments of syntactic functions. Empirical results demonstrate the effectiveness of both methods.

7 Limitations

In this section, we discuss the limitations of our methods and point out promising future directions and challenges. In our experiments, we observe MET does provide substantial improvements on SCAN and COGS by injecting a certain amount of ME bias into the models. Nonetheless, the effect of ME bias achieved by MET is not equal to the ME bias of humans. MET regularizes the model to not generate seen outputs (or expressions) facing unseen inputs (or concepts). Human ME biases, however, will additionally let humans assign two different unseen expressions (or outputs) to two different unseen concepts (or inputs). To summa-

[5]Our work was first submitted to ACL Rolling Review on March 15, 2022.

rize, MET achieves ME bias on the trained domain, which already leads to substantial improvement on SCAN and COGS. Future work should explore directions to achieve generalizable ME bias, which may lead to a further breakthrough in this task.

Additionally, as discussed in Sec. 4.2, one remaining challenge of our models is to improve generalization to novel structural arguments, where all the models achieve close to zero accuracy. Similar to the trends on other datasets, the extremely low performance of the baselines makes it hard for our deductive method MET to improve. On the other hand, our prim2primx data augmentation has the potential to improve similar problems, as reflected in the SCAN *Jump* results. However, the success depends on automatically identifying *lexicon*-level arguments. In contrast, automatically identifying any *phrase*-level structures is still an open challenge, and we leave this for future work.

8 Ethical Considerations

The methods proposed in this paper aim to improve the compositional generalization ability of the models. Progress in this direction can lead to more systematic and predictable behavior of neural models in downstream applications. In this work, improvements are evaluated on two highly-controlled datasets, SCAN (Lake and Baroni, 2018) and COGS (Kim and Linzen, 2020) as proof-of-concept experiments. We also point out potential challenges and directions to apply our methods in practice in Sec. 7.

Acknowledgements

We thank the reviewers for their helpful comments. This work was supported by ONR Grant N00014-18-1-2871, NSF-CAREER Award 1846185, and DARPA MCS Grant N66001-19-2-4031. The views are those of the authors and not of the funding agency.

References

Ekin Akyürek, Afra Feyza Akyürek, and Jacob Andreas. 2021. Learning to recombine and resample data for compositional generalization. In *International Conference on Learning Representations*.

Ekin Akyurek and Jacob Andreas. 2021. Lexicon learning for few shot sequence modeling. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4934–4946, Online. Association for Computational Linguistics.

Ekin Akyürek and Jacob Andreas. 2022. Compositionality as lexical symmetry. *arXiv preprint arXiv:2201.12926*.

Jacob Andreas. 2020. Good-enough compositional data augmentation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7556–7566, Online. Association for Computational Linguistics.

David P Ausubel. 1963. The psychology of meaningful verbal learning.

Jasmijn Bastings, Marco Baroni, Jason Weston, Kyunghyun Cho, and Douwe Kiela. 2018. Jump to better conclusions: SCAN both left and right. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 47–55, Brussels, Belgium. Association for Computational Linguistics.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.".

Philémon Brakel and Stefan Frank. 2009. Strong systematicity in sentence processing by simple recurrent networks. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 31.

Noam Chomsky. 1957. *Syntactic structures*. De Gruyter Mouton.

Henry Conklin, Bailin Wang, Kenny Smith, and Ivan Titov. 2021. Meta-learning to compositionally generalize. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3322–3335, Online. Association for Computational Linguistics.

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.

Roberto Dessì and Marco Baroni. 2019. CNNs found to jump around more skillfully than RNNs: Compositional generalization in seq2seq convolutional networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3919–3923, Florence, Italy. Association for Computational Linguistics.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR.

Daniel Furrer, Marc van Zee, Nathan Scales, and Nathanael Schärli. 2020. Compositional generalization in semantic parsing: Pre-training vs. specialized architectures. *arXiv preprint arXiv:2007.08970*.

Kanishk Gandhi and Brenden M Lake. 2020. Mutual exclusivity as a challenge for deep neural networks. *Advances in Neural Information Processing Systems*, 33:14182–14192.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jonathan Gordon, David Lopez-Paz, Marco Baroni, and Diane Bouchacourt. 2020. Permutation equivariant models for compositional generalization in language. In *International Conference on Learning Representations*.

Kristina Gulordava, Thomas Brochhagen, and Gemma Boleda. 2020. Deep daxes: Mutual exclusivity arises through both learning biases and pragmatic strategies in neural networks. In *Proceedings of the 42th Annual Meeting of the Cognitive Science Society - Developing a Mind: Learning in Humans, Animals, and Machines, CogSci 2020, virtual, July 29 - August 1, 2020*. cognitivesciencesociety.org.

Yinuo Guo, Hualei Zhu, Zeqi Lin, Bei Chen, Jian-Guang Lou, and Dongmei Zhang. 2021. Revisiting iterative back-translation from the perspective of compositional generalization. In *AAAI*.

Jonathan Herzig, Peter Shaw, Ming-Wei Chang, Kelvin Guu, Panupong Pasupat, and Yuan Zhang. 2021. Unlocking compositional generalization in pre-trained models using intermediate representations. *arXiv preprint arXiv:2104.07478*.

Yichen Jiang and Mohit Bansal. 2021. Inducing transformer's compositional generalization ability via auxiliary sequence prediction tasks. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6253–6265, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ákos Kádár, Afra Alishahi, and Grzegorz Chrupała. 2015. Learning word meanings from images of natural scenes. *Traitement Automatique des Langues*, 55(3).

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.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada. Association for Computational Linguistics.

Brenden Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International Conference on Machine Learning*, 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.

Brenden M. Lake, Tal Linzen, and Marco Baroni. 2019. Human few-shot learning of compositional instructions. In *Proceedings of the 41th Annual Meeting of the Cognitive Science Society, CogSci 2019: Creativity + Cognition + Computation, Montreal, Canada, July 24-27, 2019*, pages 611–617. cognitivesciencesociety.org.

Angeliki Lazaridou, Grzegorz Chrupała, Raquel Fernández, and Marco Baroni. 2016. Multimodal semantic learning from child-directed input. In *Knight K, Nenkova A, Rambow O, editors. Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies; 2016 Jun 12-17; San Diegio, California. Stroudsburg (PA): Association for Computational Linguistics; 2016. p. 387–92*. ACL (Association for Computational Linguistics).

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.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Adam Liška, Germán Kruszewski, and Marco Baroni. 2018. Memorize or generalize? searching for a compositional rnn in a haystack. In *AEGAP Workshop of ICML*.

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.

Qian Liu, Shengnan An, Jian-Guang Lou, Bei Chen, Zeqi Lin, Yan Gao, Bin Zhou, Nanning Zheng, and Dongmei Zhang. 2020. Compositional generalization by learning analytical expressions. In *NeurIPS*.

João Loula, Marco Baroni, and Brenden Lake. 2018. Rearranging the familiar: Testing compositional generalization in recurrent networks. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 108–114, Brussels, Belgium. Association for Computational Linguistics.

Ellen M Markman and Gwyn F Wachtel. 1988. Children's use of mutual exclusivity to constrain the meanings of words. *Cognitive psychology*, 20(2):121–157.

Richard Montague. 1970. Universal grammar. *1974*, pages 222–46.

Maxwell I Nye, Armando Solar-Lezama, Joshua B Tenenbaum, and Brenden M Lake. 2020. Learning compositional rules via neural program synthesis. In *NeurIPS*.

Inbar Oren, Jonathan Herzig, Nitish Gupta, Matt Gardner, and Jonathan Berant. 2020. Improving compositional generalization in semantic parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2482–2495, Online. Association for Computational Linguistics.

Arkil Patel, Satwik Bhattamishra, Phil Blunsom, and Navin Goyal. 2022. Revisiting the compositional generalization abilities of neural sequence models. In *Proceedings of ACL*.

Linlu Qiu, Peter Shaw, Panupong Pasupat, Paweł Krzysztof Nowak, Tal Linzen, Fei Sha, and Kristina Toutanova. 2021. Improving compositional generalization with latent structure and data augmentation. *arXiv preprint arXiv:2112.07610*.

Jacob Russin, Jason Jo, Randall O'Reilly, and Yoshua Bengio. 2020. Compositional generalization by factorizing alignment and translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 313–327, Online. Association for Computational Linguistics.

Peter Shaw, Ming-Wei Chang, Panupong Pasupat, and Kristina Toutanova. 2021. Compositional generalization and natural language variation: Can a semantic parsing approach handle both? In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 922–938, Online. Association for Computational Linguistics.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468, New Orleans, Louisiana. Association for Computational Linguistics.

Ning Shi, Boxin Wang, Wei Wang, Xiangyu Liu, Rong Zhang, Hui Xue, Xinbing Wang, and Zhouhan Lin. 2022. From scan to real data: Systematic generalization via meaningful learning. In *BlackboxNLP Workshop at EMNLP*.

Karl Stratos. 2019. Mutual information maximization for simple and accurate part-of-speech induction. In *Proceedings of NAACL-HLT*, pages 1095–1104.

Scott Thornbury. 1999. *How to teach grammar*, volume 3. Longman Harlow.

Dmitry Tsarkov, Tibor Tihon, Nathan Scales, Nikola Momchev, Danila Sinopalnikov, and Nathanael Schärli. 2021. *-cfq: Analyzing the scalability of machine learning on a compositional task. In *AAAI*.

Gail Weiss, Yoav Goldberg, and Eran Yahav. 2018. On the practical computational power of finite precision RNNs for language recognition. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 740–745, Melbourne, Australia. Association for Computational Linguistics.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. 2019. Neural text generation with unlikelihood training. In *International Conference on Learning Representations*.

Paul Rodriguez Janet Wiles. 1998. Recurrent neural networks can learn to implement symbolsensitive counting. *Advances in Neural Information Processing Systems*, 10:87.

Francis CK Wong and William SY Wang. 2007. Generalisation towards combinatorial productivity in language acquisition by simple recurrent networks. In *2007 International Conference on Integration of Knowledge Intensive Multi-Agent Systems*, pages 139–144. IEEE.

Hao Zheng and Mirella Lapata. 2021. Compositional generalization via semantic tagging. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1022–1032, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Hao Zheng and Mirella Lapata. 2022. Disentangled sequence to sequence learning for compositional generalization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4256–4268, Dublin, Ireland. Association for Computational Linguistics.

Appendix

A Methods

A.1 Diversify Lexical Arguments with Primitive Augmentation

Building a Lexicon. Enforcing the two conditions in Eqn. 1 on the SCAN *Jump* task would return 6 pairs of primitives, e.g., (*run*, RUN), (*walk*, WK), etc. However, for other tasks with a larger vocabulary, a word's different surface forms are often parsed to the same output meaning. For example, in COGS, both "*paint*" and "*painted*" are mapped to "paint" in the output form. Therefore, we use the "*no-winner*" condition (Akyurek and Andreas, 2021) that allows an M-to-one mapping if v is not necessary for w as long as $M <$ threshold value ψ.

$$
\begin{aligned}
\text{NoWinner}(w) &= \nexists v, \text{suff}(v, w) \wedge \text{ness}(v, w) \\
\text{IsPrim}(v, w) &= \text{suff}(v, w) \wedge \\
&\quad (\text{ness}(v, w) \vee \text{NoWinner}(w))
\end{aligned}
$$

Finally, we delete words above a certain frequency and words below another frequency from the lexicon. This frequency thresholds is decided based on dev-set tuning results.

Compared to the previous compositional data augmentation methods (Andreas, 2020; Akyürek et al., 2021) that substitute words with other semantically equivalent words from the same lexicon, our method has two main advantages: (1) it circumvents the difficult task of finding semantically equivalent words that share some common environment and creating templates that can be filled with these words. (2) because we are not replacing the primitive with another word from the existing lexicon (e.g., *jump*), the augmented data will not expose/reveal any novel compositions ("*jump around left*") in the test set, and thus still preserve the compositional generalization challenge in the original task. Another side advantage of creating new primitives instead of swapping with one from the original lexicon is that we can train the model to generalize to future new primitives (e.g., dax) simply by adding "*dax*↦DAX" to the training set without the need to rerun any data-augmentation procedures.

Algorithm 1 prim2primX Data Augmentation

```
Require: Training data $D$
Require: All primitives $P$
Require: Number of new primitives $n$
Ensure: Augmented data $D_a$
  for $(x, y) \in D$ do
      $D_a = D_a \cup (x, y)$
      for token $x_t$ in $x$ do
          $x' \leftarrow x, y' \leftarrow y$
          $y_t \leftarrow P[x_t]$
          if $t \in P$ then
              Sample $s$ from $\{0, 1, ..., n\}$
              if $s = 0$ then break
              end if
              Replace all $x_t$ in $x'$ with $x_t s$
              Replace all $y_t$ in $y'$ with $y_t s$
          end if
      end for
      $D_a = D_a \cup (x', y')$
  end for
```

B Experimental Details

B.1 SCAN Dataset

The SCAN dataset (Lake and Baroni, 2018) consists of natural language commands paired with action sequences. Each sub-command is made of three types of words: action primitive ("*walk, jump, look, run, turn*"), direction primitive ("*left, right*"), and functional words ("*opposite, around, twice, thrice*"), and can be connected with another sub-command via a function ("*and, after*"). Table 9 describes all syntactic functions of SCAN.

Jump evaluates the model's ability to generalize compositions of syntactic functions across different lexical (primitive) arguments. The training set consists of the primitive command "*jump*" on its own, all other primitives, and compound commands *without* "*jump*" (e.g., "*walk around left*"); the test set contains compound commands with "*jump*" (e.g., "*jump around left*").

Around Right (Loula et al., 2018) puts all commands containing templates of the form "*[primitive] around right*" in the test set, with the remaining examples in the training set.

MCD splits are created to maximize the output compound divergence while guaranteeing a small atom divergence between train and test sets (Keysers et al., 2020). For example, the training and dev sets of MCD1 have a similar distribution of individual words to ensure minimal atom divergence. However, the training set does not contain compounds "*[primitive] around left twice*", which only appear in the dev and test sets. These splits require

Function f	Examples		
	Input	**Program**	**Output**
$\text{Identity}(x) = x$	walk	Identity(walk)	WK
$\text{Rev}(x_1, x_2) = x_2 + x_1$	walk left	Rev(walk, left)	TL WK
$\text{Oppo}(x_1, x_2) = x_2 + x_2 + x_1$	walk **opposite** left	Oppo(walk, left)	TL TL WK
$\text{Around}(x_1, x_2) = (x_2 + x_1) * 4$	walk **around** left	Around(walk, left)	TL WK TL WK TL WK TL WK
$\text{Twice}(x) = x * 2$	walk **twice**	Twice(walk)	WK WK
	walk opposite left **twice**	Twice(Oppo(walk))	TL TL WK TL TL WK
$\text{Thrice}(x) = x * 3$	walk **thrice**	Thrice(walk)	WK WK WK
	walk opposite left **thrice**	Thrice(Oppo(walk))	TL TL WK TL TL WK TL TL WK
$\text{And}(x_1, x_2) = x_1 + x_2$	walk left **and** run	And(Rev(walk, left), run)	TL WK RUN
$\text{After}(x_1, x_2) = x_2 + x_1$	walk left **after** run	After(Rev(walk, left), run)	RUN TL WK

Table 9: All syntactic functions in SCAN (Lake and Baroni, 2018) and examples of applying these functions to primitive and structural arguments. "$x_1 + x_2$" means concatenating x_1 and x_2 at the output side and $x * 2$ means replicating x twice.

a higher level of compositionality than recognizing the syntactic equivalence of primitives: the models must be able to (1) understand the underlying symbolic functions "x twice$\mapsto x\ x$" from training examples like "*jump left twice*" and master the semantics of "*jump around left*" from examples like "*jump around left thrice*"; (2) compositionally apply the function "twice" to a novel argument "*jump around left*" in the dev and test sets.

B.2 COGS Dataset

The COGS dataset (Kim and Linzen, 2020) requires parsing a diverse set of natural language sentences into their corresponding logical forms based on lambda calculus to accurately reflect the semantic representation of the natural sentence. COGS raises five different systematic generalization challenges in its test set: (1) novel combination of familiar primitives and syntactic functions; (2) novel combination of modified phrases and syntactic functions; (3) sentences with deeper recursion; (4) sentences with alternative verb argument structures (e.g., active-passive), and (5) novel identity of a verb (e.g., unaccusative and unergative). Challenges 1, 4, and 5 require generalizing a syntactic function to a lexical argument which was never associated with this function during training, while challenges 2 and 3 require generalizing to unseen structural arguments.

B.3 Experimental Setup

Baselines We report results for both LSTMs and Transformers in our experiments. Our LSTM hyper-parameters are adopted from Akyurek and Andreas (2021), with 2 layers in both encoder and decoder, a hidden dimension size of 512, and a dropout rate of 0.4. Our Transformer hyperparameters and configurations are set following the findings in Csordás et al. (2021). We use relative positional embedding (Shaw et al., 2018) and use 3 layers in both encoder and decoder. The hidden dimension size and the embedding size are set to 256, and the dimension of the feed-forward layer is set to 512. We use a dropout rate of 0.1 and 4 self-attention heads. The Transformer model is implemented using OpenNMT (Klein et al., 2017). Additionally, we also report the performance of the meta-learning method used in Conklin et al. (2021) as our baseline, and will be shown as "+MAML" in the result tables. Since in our preliminary experiments, we notice different baselines have a substantial influence on the performance (Csordás et al., 2021), we report the reimplemented performance of their method using the Levenshtein distance on the previously described LSTM and Transformer baselines for a fair comparison with other approaches.

Training Details Recent works (Csordás et al., 2021; Conklin et al., 2021) observed that the in-domain development set cannot provide enough signal for doing model selection for compositional generalization. Hence, similar to Conklin et al. (2021), we split the original test set into a new development set that contains 10% of the examples and a new test set containing the rest 90%. We select the checkpoint with the best accuracy on the development set. In all our experiments, we report the mean performance in 5 runs and the corresponding standard deviation.

For all our LSTM models, we train our model using a dropout rate of 0.4, a peak learning rate of 1.0 with the Noam learning rate scheduling. The baseline model is trained using a batch size of 512, and for 8000 steps with 4000 warm-up steps on

Model	Performance
Transformer	62.84±9.54
+ MET	100.00±0.00
+ MET-Meta	100.00±0.00

Table 10: The performance of the Transformer baseline and MET on the synthetic task described in Sec. C.1

.

Model	*Jump* *	COGS
Transformer	62.80±15.61	75.90±2.34
+ Avg-Word Unlike	1.32±0.43	6.16±1.39
+ Min-Word Unlike	59.97±8.60	**80.58**±0.78
+ Sent Unlike	60.68±8.96	80.52±1.13
+ Meta Sent Unlike	**73.30**±21.04	79.44±1.39

Table 11: Unlikelihood loss ablation, reported as dev accuracy on SCAN *Jump* and COGS. Models for *Jump* are trained on augmented data.

the SCAN datasets. For the primitive augmentation datasets, we notice the model needs a longer time to converge since the dataset is multiple times larger, so we train those models for 40000 steps with 4000 warm-up steps. For all our Transformer models, we train our model using a dropout rate of 0.1, a peak learning rate of 2.0 with the Noam learning rate scheduling. The model is trained using a batch size of 128, and for 50000 steps with 5000 warm-up steps in all the experiments. All our models can be trained on a single NVIDIA Titan Xp GPU.

C Additional Results

C.1 Encouraging Mutual Exclusivity in a Synthetic Experiment

In this experiment, we design a highly-controlled synthetic experiment to demonstrate how MET encourages mutual exclusivity in the model. Our experiment design is adapted from the toy experiment design in Zheng and Lapata (2022).

We construct the dataset using two bijections: f_1 between a source vocabulary S_1 and a target vocabulary T_1, and f_2 between vocabulary S_2 and T_2. The set S_1 can be further split to two disjoint sets S_1^{train} and S_1^{gen}, each mapping to T_1^{train} and T_1^{gen} respectively.

Following Zheng and Lapata (2022), we construct our training set so the model can exploit some spurious correlations between the input and the output. Specifically, the training set will contain two types of examples. For the first type of examples, the input only contain one token $s_1 \in S_1$ the output is $f_1(s_1)$. For the second type of examples, the input sequence contains two tokens 's_1^{train}, s_2', where $s_1^{train} \in S_1^{train}$ and $s_2 \in S_2$. The corresponding output is '$f_1(s_1^{train}), f_2(s_2)$'.

All the testing examples will look similar to the second type and contain two tokens, s_1 and s_2, where $s_1 \in S_1$ and $s_2 \in S_2$. Crucially, the main difficulty of this testing set lies in predicting the correct outputs for inputs that **never** appears in the training set, i.e. inputs containing s_1^{gen} and s_2, while s_1^{gen} never appears in the second type of examples in the training set.

The performance of the Transformer baseline is shown in Table 10. We can see that despite the simplicity of this task, the Transformer baseline struggles to perform well on the novel examples, only reaching 62.80 accuracy. A common failure case for the baselines is to only predict $f_1(s_1^{gen})$ for the input sequence 's_1^{gen}, s_2' and completely ignores the second input token. This is because during training, s_1^{gen} only appears in sentences with total length 1, and the model can exploit this spurious pattern. By applying the MET framework, we can prevent this incorrect behavior, since the output $f_1(s_1^{gen})$ should only be matched with the input s_1^{gen} and should not be paired with any other input. The empirical improvements on this task confirm the effectiveness of MET. From Table 10, we can see using both MET or MET-Meta can make the model reach 100% accuracy and completely solves this task.

C.2 Ablation Experiments on the Implementation of the Unlikelihood Loss

Notably, the unlikelihood loss $\mathcal{L}_{\text{UL}}$ implementation used in MET is different from the loss used in Welleck et al. (2019), as the original unlikelihood is an average over all possible locations, while the unlikelihood loss in Eq. 3.1 is operated at the sentence level. We make this design choice since our unlikelihood is not used to penalize the generation of each token (which is the target in Welleck et al. (2019)), but to penalize the generation of the whole sentence, in which only several key tokens are wrong and should be penalized. The sentence-level unlikelihood loss can achieve this by only maximizing the unlikelihood on a few words, while keeping the likelihood on most words unchanged. Our ablation experiments below also verify the advantage of this design choice.

Empirical Comparison of Different Loss Functions In Table 11, we compare the effect of different unlikelihood loss variants. On both SCAN

Jump and COGS, our sentence-level variant (Sent Unlike) is substantially better than the original variant used in Welleck et al. (2019) that averages unlikelihood loss at the word level (Avg-Word Unlike), as the penalization effect of the latter method is too strong and lead to a very bad final performance. In our preliminary experiments, we have also tried to use a smaller weight coefficient for the Avg-Word Unlike variant. While using a smaller coefficient is beneficial, the Sent Unlike variant still shows an advantage. We also experimented with another variant where we still compute the unlikelihood loss at the word level, but use a min-pooling operation to get the final loss, which achieves a similar effect as our Sent Unlike loss since the penalization will only apply to a certain token. From Table 11, we can see this variant shows similar performance to the Sent Unlike, further verifying our hypothesis about why Sent Unlike is preferable to the original variant in Welleck et al. (2019) in our experiments.

D Examples for the effect of prim2primX on COGS

Given a training example "*A rose was helped by a dog*", our prim2primX data augmentation will first identifies "*rose*", "*helped*", and "*dog*" as primitives,and then randomly swap some of these primitives with their mutated forms to create new training examples (e.g., "*A **rose1** was **helped0** by a dog.*"). The quantitative results on COGS are in Table 2 in the main paper. Just by using the prim2primX data augmentation, we can improve the Transformer baseline from 76.14 % accuracy to 80.07% accuracy, showing a 3.93% absolute improvement from the model trained with original data. MET further boosts the accuracy to 81.12%, again demonstrating that *deductive* MET method can provide complementary performance gain on top of *inductive* data augmentation.