Institute for Natural Language Processing
Universität Stuttgart

Master thesis in Computational Linguistics

Compositional Generalization through Learning to In-context Learn

Sungjun Han

Supervised by:	Prof. Dr. Sebastian Pado
Examiner:	Prof. Dr. Sebastian Pado

Start Date:	31.10.2022
Submission Date:	19.05.2023

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst habe und dabei keine andere als die angegebene Literatur verwendet habe. Alle Zitate und sinngemäßen Entlehnungen sind als solche unter genauer Angabe der Quelle gekennzeichnet. Die eingereichte Arbeit ist weder vollständig noch in wesentlichen Teilen Gegenstand eines anderen Prüfungsverfahrens gewesen. Sie ist weder vollständig noch in Teilen bereits veröffentlicht. Die beigefügte elektronische Version stimmt mit dem Druckexemplar überein.

(Sungjun Han, 19.05.2023)

Abstract

Compositional generalization refers to the ability to generalize to new distributions by combining the familiar concepts in a novel manner. The significance of compositional generalization has been consistently highlighted in the field of natural language processing. Most recently, large language models have shown remarkable ability for compositional generalization by learning from a few demonstrative examples without any changes to its weights. Then, how is this ability to learn from a few examples, known as in-context learning, related to their ability to compositionally generalize?

This thesis studies this research question by hypothesizing that the two abilities are closely interconnected. In order to learn from experience, the model needs to accurately capture the underlying latent structure that are shared between the demonstrative examples and this representation can be useful for compositional generalization.

We devise a novel memory-based meta-learning method that can directly train Transformers for in-context learning and test for compositional generalization. It trains the model on different online learning trajectories of a sequence to sequence dataset.

The experimental results show that the in-context learning model trained with our meta-learning method can improve the performance of the Transformer model on several difficult datasets that test for compositional generalization, demonstrating that the two abilities are indeed tightly coupled. Also, we show that in-context learning can arise when the model has the opportunity to memorize the data in its weights. Lastly, we elucidate the sensitivity of the Transformer in the generalization problems implicit in different online learning trajectories of the dataset and how this determines the compositional generalization capability of the model.

Contents

1 Introduction **1**

2 Background **5**
2.1 Compositional Generalization . . . 5
2.1.1 Types of Compositionality . . . 5
2.1.2 Compositional Generalization as Out-of-Distribution Generalization . 6
2.1.3 Testing for Compositionality . . . 7
2.1.4 Designing the Compositional Split . . . 7
2.2 Natural Language Processing with Deep Learning . . . 9
2.2.1 Problem Setup . . . 9
2.2.2 Deep Leaning for Sequence Modelling . . . 9
2.2.3 Transformer and Positional Embeddings . . . 10
2.3 Compositional Geneneralization in Deep Neural Networks . . . 10
2.3.1 Lack of Compositionality in DNNs . . . 11
2.3.2 Emergence of Systematicity by the Property of Data Distribution . . 12
2.4 Memory-based Meta Learning . . . 14
2.4.1 Problem Setup . . . 15
2.4.2 Important Considerations regarding Meta-memory learning . . . 16

3 Related Works **19**
3.1 Compositional Generalization . . . 19
3.1.1 Debate on Compositionality in Deep Learning Models . . . 19
3.1.2 Inductive Priors for Compositional Generalization . . . 20
3.2 Meta-Learning . . . 22
3.2.1 General Overview of Meta-learning . . . 22
3.2.2 Meta-Learning for NLP . . . 23
3.2.3 In-context Learning in DNNs . . . 24

4 Memory-based Meta-Learning for Compositional Generalization **27**
4.1 Generating the Task Distribution by Permutation of Samples . . . 27
4.2 Generating the Task Distribution by Permutation of Labels . . . 29
4.3 Implementation . . . 30
4.4 Evaluation . . . 31
4.5 Theoretical Limit on Generalization Capability . . . 32

5 Experiments and Results **34**
5.1 Datasets . . . 34
5.1.1 SCAN . . . 34
5.1.2 AA and AATT . . . 35
5.2 Implementation Details . . . 36
5.2.1 Preprocessing . . . 36
5.2.2 Model . . . 36
5.2.3 Task Distribution . . . 37

5.2.4 Evaluation . . . 37
5.3 Experiment 1 . . . 38
5.3.1 Experimental Setup . . . 38
5.3.2 Results . . . 39
5.4 Experiment 2 . . . 40
5.4.1 Experimental Setup . . . 41
5.4.2 Results . . . 42
5.5 Experiment 3 . . . 44
5.5.1 Experimental Setup . . . 44
5.5.2 Results . . . 45
5.6 Experiment 4 . . . 47
5.6.1 Experimental Setup . . . 48
5.6.2 Results . . . 49

6 Conclusion **52**

A Datasets for Testing Compositional Generalization **54**
A.1 SCAN . . . 54
A.2 PCFG . . . 54
A.3 COGS . . . 55
A.4 CFQ . . . 56

B Test-score Trajectories without Smoothing **58**
B.1 Experiment 1 . . . 58
B.2 Experiment 2 . . . 59
B.3 Experiment 3 . . . 60

List of Figures **63**

List of Tables **65**

References **69**

1 Introduction

For humans, knowledge of how a particular concept is used implies knowledge of its use in other structurally related concepts. Furthermore, we have the ability to recursively combine them with other concepts in a creative manner, affording us with an "infinite use of finite means" (Chomsky, 1965). These capabilities arise out of the ability to combine the atomic parts in reoccurring structures in a novel manner and is known altogether as compositional or systematic generalization. The importance of compositional generalization in building artificial intelligence (AI) has been emphasized for several decades (Smolensky, 1988; Fodor and Pylyshyn, 1988; Marcus, 2001; McClelland et al., 2010; Calvo and Symons, 2014; Lake et al., 2018). However, there exists a significant debate between researchers on *how* systematic generalization could be implemented in an AI model. Especially, the historical debate on compositionality centered around whether the *connectionist* architecture (i.e. deep learning) could even be capable of systematicity.

With the recent remarkable advances in deep learning (Schmidhuber, 2015), this particular kind of out-of-distribution (OOD) generalization capability gained renewed attention by AI research communities using synthetic sequence to sequence datasets (Lake and Baroni, 2017; Hupkes et al., 2020; Keysers et al., 2020; Shaw et al., 2021; Russin et al., 2019). In these approaches, entire datasets are split into the train and test set in a systematic and non-i.i.d fashion where by the model can successfully generalize only through uncovering the true latent structure of the data distribution. For example, upon being familiarized with a new verb *dax* during training, the model is tested on the novel uses of the word with the other familiar words such as *dax twice*. The results showed that the modern deep learning architectures such as LSTM (Hochreiter and Schmidhuber, 1997), CNN (LeCun et al., 1990), and Transformer (Vaswani et al., 2017) seem to be not capable of compositional generalization, proving a fatal flaw in these models.

However, the detailed picture of the systematicity of modern deep learning architectures turned out to be much more complex. Reminiscent of the famous "more is different" argument (Anderson, 1972; McClelland et al., 2010), many subsequent works have noted an emergence of systematicity in these systems upon an increase in diversity and scale of the data (Yu et al., 2018; Hill et al., 2020; Patel et al., 2022). Most recently, the large language models (LLMs) using the Transformer layers have also shown emergent *in-context learning* capability (Brown et al., 2020; Chowdhery et al., 2022) where by it can learn from a few examples of new concepts provided in the context without any updates to the weights (Garg et al., 2022). Especially when equipped with special prompting methods, these models are shown to be capable of systematic generalization in a wide range of problems through in-context learning (Wei et al., 2022b; Zhou et al., 2023; Lampinen et al., 2022). Then, *is the in-context learning ability of LLMs related to the ability to compositionally generalize?*

We hypothesize that the ability to in-context learn directly implies the ability to compositionally generalize. The model that is able to learn from experience without any changes to the weights, must be able to extract and represent the true shared latent structure governing the examples. This structural representation could then be utilized to generalize to other novel but structurally equivalent examples. For example, our ability to understand "*dax*

twice" can be framed as a generalization of our knowledge of past uses of "*twice*" in other contexts such as "*eat twice*" or "*bark twice*".

However, the in-context learning of LLMs are *emergent*, meaning that they were not directly trained for such capability. Hence, we refer to the memory-based meta-learning approach (Hochreiter et al., 2001; Ortega et al., 2019) which provides another way to train for in-context learning using minimal inductive bias. Meta-learning (Schmidhuber et al., 1996; Bengio et al., 1991; Thrun and Pratt, 1998; Hochreiter et al., 2001) is a "learning to learn" approach which trains the model on a distribution of generalization problems to discover a general strategy that can be successful on all the tasks. Meta-memory learning generates a distribution of sequence trajectories where the model needs to improve the loss on each trajectory by predicting the next label given the true ground label from the previous time step. It has been successfully applied to few-shot learning problems where in-context learning ability can be beneficial (Santoro et al., 2016; Duan et al., 2017; Wang et al., 2017). For example, a maze-solver capable of remembering its past moves in a run is trained on many different but similarly structured mazes. In order to be successful on *all* the mazes, the model develops a general policy for solving mazes rather than memorizing the solution of each individual maze.

In this thesis, we study our hypothesis by devising a new memory-based meta-learning training regime that trains for in-context learning and testing the resulting model's capability to compositionally generalize. Given a sequence to sequence dataset testing for compositional generalization, we present the dataset to the model in an online manner. This way, the model is forced to correctly organize the knowledge of previously processed input-output pairs to successfully generalize on the next example as it only sees each example exactly once in the sequence.

Thus, this approach directly trains the model for generalization of novel but related examples. Since the dataset does not have a natural ordering, we can generate different ordering of dataset each with distinct generalization problems. In other words, we define our task distribution for meta-learning through permutations of the dataset order.

In addition, we can also further increase the size of our distribution of tasks by defining different ways of representing the vocabulary. We do this by defining different permutations of the vocabulary labels which results in the surface forms of the words being shuffled and swapped. However, the underlying linguistic structure of the dataset would still be the same. With this addition, the model is no longer able to memorize the examples in its weights (i.e. in-weights learning) as multiple labels can be assigned to each word. Hence, the model has a strong signal to discover the true linguistic structure from the past examples to generalize to the future examples.

We extensively study our approach on several difficult compositional generalization splits of the sequence to sequence dataset called SCAN (Lake and Baroni, 2017) using the Transformer architecture. We find that Transformer can learn from the generalization problems present in different online learning trajectories of the dataset and can compositionally generalize when conditioned on random training trajectories.

Furthermore, our findings demonstrate that the model can learn to in-context learn even in

the presence of in-weights learning opportunities. Also, the in-context learning ability can generalize to an extent to learning from novel sequences. Another contribution of our work is the demonstration that the Transformer model is sensitive to the few-shot learning problems implicit in the generated meta-tasks and this determines the model's compositional generalization capability. Specifically, the important role the positional embedding of Transformer plays in this phenomenon is elucidated.

The thesis is structured as follows: in Chapter 2, we provide relevant background information needed for understanding our work and contributions on the subject of compositional generalization. We first discuss how the compositionality is defined and how this can be tested on a deep learning model (Section 2.1). Next, we familiarize the readers with the major results on the lack of compositionality in deep learning models as well as the conflicting results which shows the emergence of compositionality in the same model (Section 2.3). Then, the memory-based meta-learning framework is presented and the results on the emergence of compositionality is interpreted through this framework (Section 2.4). In Chapter 3, we cover related works in compositional generalization (Section 3.1) and meta-learning (Section 3.2) to demonstrate in what context our work sets itself apart from previous research. Then, our memory-based meta-learning framework for compositional generalization is presented in Chapter 4 with details on how they can be implemented and how they can be evaluated for zero-shot generalization along with the method's theoretical limitations. In Chapter 5, we empirically evaluate our method through a series of experiments on different compositional generalization splits of SCAN. Finally, we provide a comprehensive summary of our findings and engage in a discussion regarding potential avenues for future research in Chatper 6.

2 Background

This section explains important background information for understanding the thesis. In Section 2.1, we present the problem of compositional generalization and how it is a type of out-of-distribution generalization. We also explain how it can be tested on deep neural networks and cover the relevant methods for testing. Next, we briefly introduce the sequence prediction problem of natural language processsing along with established deep learning architectures for sequence modelling in Section 2.2. Then, Section 2.3 covers the topic of lack of compositionality on modern deep learning networks. Lastly, memory-based meta-learning is introduced formally in Section 2.4 together with considerations that are important for understanding this thesis.

2.1 Compositional Generalization

There is no consensus on an exact rigorous definition of compositional generalization and it can be understood as a rough set of several generalization capabilities inspired by how humans generalize linguistically (Fodor and Pylyshyn, 1988; Marcus, 2001; Lake et al., 2018; Schulz et al., 2016). They can be commonly explained by the principle of compositionality (Chomsky, 1965; Cann, 1993) where the meaning of a complex expression is composed of two distinct parts, the constituent parts that make up the expression and the structure that combines them. Adhering to this principle make way for different types of generalization capability which we detail in Section 2.1.1.

It is also possible to treat compositional generalization under the machine learning perspective (Section 2.1.2). Under this perspective, it is viewed as a type of out-of-distribution generalization with covariate shift.

2.1.1 Types of Compositionality

There are two kinds of fundamental compositional generalization problems: *systematicity* and *productivity*, which are the main drivers of human's creative use of language allowing us to describe and understand novel ideas of arbitrary complexity.

Systematicity refers to the ability to recombine known parts of familiar sequences into a sequence of novel combinations. In comprehension and production of natural language, the understanding of a certain sentence implies the understanding of other structurally related sentences. For example, following the famous example of Fodor and Pylyshyn (1988), understanding the meaning of "John loves Mary" implies the capability to understand "Mary loves John". To give another example, a visually grounded concept such as color can be generalized in novel combinations with other objects where one might be able to locate a "purple banana" upon instruction even though such a combination has never been experienced before. In essence, systematicity implies representing the world as a composition of atomic parts and the structure behind the interaction of these atoms such that more complex concepts can be represented by variably composing the atoms.

Productivity is an ability to combine the atoms into compounds recursively arbitrary number of times. For example, in English it is possible to use the word "that" to produce

a sentence which can be extended repeatedly for an infinite number of times such as "The boy ate the snake that ate the bear that ..." . Hence, it can be considered as a form of structural generalization by novelly combining familiar structures. Though there is probably some physical limit to the human linguistic productivity due our finite memory, the language in principle is infinite (Chomsky, 1965).

Some works have made proposals to extend systematicity and productivity in defining compositional generalization with the ability to constrain the generalizations. (Hupkes et al., 2020) proposed the concepts such as *substitutivity* and *overgeneralization*. Generalization of linguistic concepts can be constrained semantically or through social customs. *Substitutivity* captures the semantic aspect of linguistic constraints by checking how substituable synonyms are. *Overgeneralization* measures the level of systematic application of rules to the normative exceptions. It follows from the known evidence on how humans overgeneralize grammatical rules given inadequate information to constrain them (Goldberg, 2019).

In this work, we focus on systematicity and productivity in studying compositional generalization and do not consider the constraints on generalization.

2.1.2 Compositional Generalization as Out-of-Distribution Generalization

The machine learning perspective on compositional generalization is to consider it as a type of out-of-distribution (OOD) generalization. In order present this perspective, we define the supervised learning problem usually employed in training machine learning models including the modern deep neural networks.

Supervised Learning and OOD. Given a feature space $\mathcal{X}$ and the label space $\mathcal{Y}$, the problem of supervised learning aims to find a function $f_\theta^* : \mathcal{X} \to \mathcal{Y}$ parameterized by θ by using the samples $\{(x_i, y_i)\}_{i=1}^N$ from the training distribution $P_{train}(\mathcal{X}, \mathcal{Y})$ that minimizes the loss function $l : \mathcal{Y} \times \mathcal{Y} \to \mathrm{R}$ on the test distribution $P_{test}(\mathcal{X}, \mathcal{Y})$. The standard supervised learning problems assume that the training and test distributions are both composed of independently and identically distributed (i.i.d) samples from a common distribution. Then by optimizing through Empirical Risk Minimization (ERM) (see Equation 1), which minimizes the average loss on the training samples, ensures successful generalization to the test distribution.

$$\mathcal{L}_{ERM} = \frac{1}{N} \sum_{i=1}^{N} l(f_\theta(x_i), y_i) \tag{1}$$

However, various studies have shown that deep neural networks trained with supervised learning are not able to systematically generalize (Lake and Baroni, 2017; Hupkes et al., 2020; Keysers et al., 2020; Kim and Linzen, 2020; Ruis et al., 2020; Bahdanau et al., 2019) (more in detail in Section 2.3.1). This is because the train and test distributions for testing systematic generalization are not i.i.d ($P_{train}(\mathcal{X}, \mathcal{Y}) \neq P_{test}(\mathcal{X}, \mathcal{Y})$). Hence, systematic generalization is a specific type of OOD generalization.

In order for a model to have a chance to generalize from the train to the test, the two distributions must be related in some way. Hence, even though $P_{train}(\mathcal{X}, \mathcal{Y})$ and $P_{test}(\mathcal{X}, \mathcal{Y})$

are different for compositional generalization, they are to be related through a covariate shift. This means that the marginal distributions $P_{train}(\mathcal{X})$ and $P_{test}(\mathcal{X})$ differ, but the conditional distribution of the labels are the same $P_{train}(\mathcal{X}) = P_{test}(\mathcal{Y}|\mathcal{X})$. Hence, the same label generation process underlie both distributions which for compositional generalization are the rules on how the concepts are combined.

2.1.3 Testing for Compositionality

Unlike the symbolic AI systems which are rule-based in nature, thus can be ensured to be compositional, deep neural networks (DNN) are black-box models that starts in a blank state but learns to tune its weights from training. They are theoretically capable of approximating all possible functions and distributions given enough resource (Hornik et al., 1989; Lu and Lu, 2020). However, this does not mean that a DNN can easily find a compositional solution through gradient descent in practice. For instance, Liška et al. (2018) found that only ∼0.1% of recurrent neural networks (RNN) (Elman, 1990) arrived at a compositional solution. This means that it depends on the data, the optimization and the amount of compute used for training for the model to reach the desired parameter space. The choice of architecture might also limit the possiblity of finding such a parameter space due to class of functions that they are able to model.

Also, since DNNs are black-box models, it is difficult to introspect the weights of a trained model to assess their systematicity. Furthermore, it is difficult to test for a specific compositional generalization (i.e. OOD generalization) capability of the model trained on natural data as we cannot control the information contained in the training data.

Hence, compositional generalization capability of DNNs are tested through an OOD problem by splitting a dataset into the train set and the test set in a systematic way. A random split of the dataset serves as a control for the model's generalization capability in the i.i.d condition. Then, the model is trained on the train set from scratch and evaluated on how well they are able to generalize on the corresponding test set (see Figure 1 for illustration). These dataset splits are called **compositional splits** as they can measure for the compositional generalization capability of the model when it is trained on the data.

Different splits can be constructed from the same dataset controlling for the different aspects of compositional generalization. In this way, similar to the principle behind random-controlled trials, other compounding factors that can possibly affect the generalization can be statistically controlled in the performance measurement.

2.1.4 Designing the Compositional Split

One must have the knowledge of the dataset's generative mechanism in order to design the compositional splits. Due to this reason, many researchers have relied on synthetic datasets of artifical languages to construct the compositional splits for both unimodal (Lake and Baroni, 2017; Dessì and Baroni, 2019; Oren et al., 2021; Hupkes et al., 2020) and grounded multi-modal settings (Ruis et al., 2020; Bahdanau et al., 2019). This way, they can have a full control of the rule-based data generative mechanism. This automatic rule-based process can also be employed for natural language instructions through the design of a generative

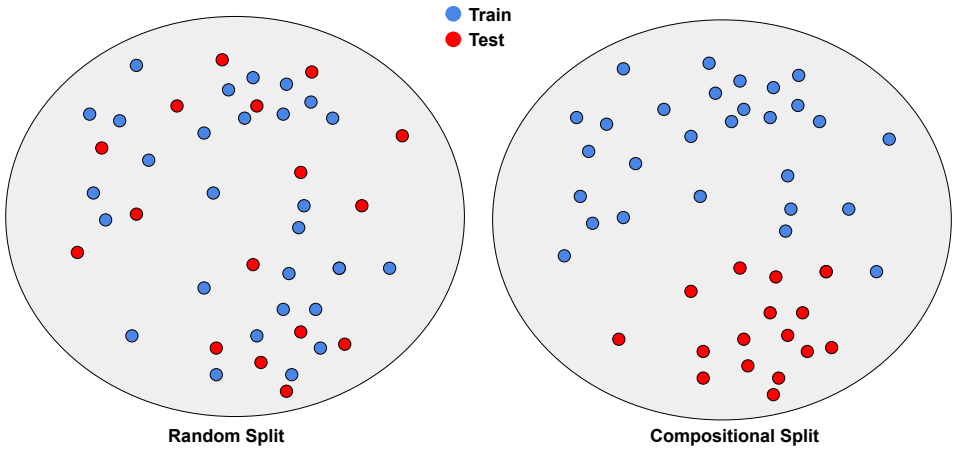


Figure 1: An illustration of a compositional split compared to a random split (i.e. i.i.d split). Compositional split divides the dataset in a non-i.i.d but systematic way.

grammar for semantic parsing (Keysers et al., 2020; Kim and Linzen, 2020; Finegan-Dollak et al., 2018; Shaw et al., 2021) or question answering (Johnson et al., 2016). When such information are not available, trained syntactic parsers or heuristics can be employed in approximating the generative structure of a natural language dataset (Agrawal et al., 2017).

Given a dataset, the process of systematically splitting dataset into into two disjoint parts is designed. Some works have relied on known types of linguistic generalization (Lake and Baroni, 2017; Kim and Linzen, 2020; Hupkes et al., 2020) such as primitive generalization (e.g. generalizing the noun in the subject role to the object role) or length/depth generalization (e.g. generalizing to longer sentences). While others have relied on the definition of systematicity by withholding some subset of possible combinations from the dataset and testing on these novel combinations (Liška et al., 2018; Hupkes et al., 2020).

Keysers et al. (2020) was first to propose a systematic way of generating the compositional split by optimizing for **Maximum Compound Divergence** (MCD). It captures the idea that compositional generalization is a type of OOD generalization. They construct the train and test distributions by making the frequency distribution of atoms similar (i.e. $P^{atom}_{train}(\mathcal{X}) \approx P^{atom}_{test}(\mathcal{X})$) and the frequency distribution of compounds to diverge (i.e. $D(P^{comp}_{train}(\mathcal{X}), P^{comp}_{test}(\mathcal{X})) > 0$). By controlling for the level of divergence between compound distributions, it becomes possible to control the difficulty of each compositional split (see Appendix A.4 for more information).

Hence, the MCD principle provides one way to capture both systematicity and productivity more generally. Under this method, systematicity can arise when some compounds containing some subset of atoms appear less frequent in the train than the test. Productivity can be considered as a divergence where the longer applications of some rules are more frequent in the test distribution than the train distribution.

2.2 Natural Language Processing with Deep Learning

In our work, we make use of the deep learning models on the problem of natural language processing (NLP). More specifically, we work on the sequence to sequence (text-to-text) problem where the objective is to convert the input sequence of text to the corresponding output sequence of text. In this section, we briefly explain the problem setup of sequence to sequence modelling (Section 2.2.1) and the established deep learning architectures and optimization framework for solving these kinds of problems (Section 2.2.2). Then, with a focus on the Transformer architecture, we briefly explain the important aspects of the presented architectures for modelling sequences (Section 2.2.3).

2.2.1 Problem Setup

The problem of natural language processing (NLP) is to model the joint probability distribution of ordered sets of tokens $P(\tau_y)$ where $\tau_y = [y_1, ..., y_m] \in \mathcal{V}^m$ and $\mathcal{V}$ is a finite alphabet (for NLP problems, $\mathcal{X}$ and $\mathcal{Y}$ are often not disjoint: $\mathcal{V} = \mathcal{X} \cup \mathcal{Y}$). It can be conditioned on an another string of tokens τ_x (i.e. sequence to sequence problem) such that $P(y_1, ..., y_m | x_1, ..., x_n)$ is computed where $\tau_x \in \mathcal{V}^n$. The standard practice is to decompose the joint probability into the conditional probabilities of tokens

$$P(y_1, ..., y_m | x_1, ..., x_n) = \prod_{i=1}^{m} P(y_i | y_{<i}, \tau_x) \tag{2}$$

by the chain-rule where $y_{<i} = [y_1, ..., y_{i-1}]$. Essentially all NLP tasks can be framed as a sequence prediction problem by modelling the conditional probabilities by concatenating τ_x and τ_y using special delimiter tokens (Radford et al., 2019). This task of modelling the conditional probability of the next token conditioned on the previous tokens is known as **language modelling**.

2.2.2 Deep Leaning for Sequence Modelling

The popular architectures used for modelling the conditional probabilities are LSTM (Hochreiter and Schmidhuber, 1997), CNN (LeCun et al., 1990), and Transformer (Vaswani et al., 2017) usually as an encoder-decoder sequence to sequence model (Sutskever et al., 2014; Bahdanau et al., 2014; Gehring et al., 2017). Transformer was originally presented as a sequence to sequence model but it can be used for language modelling by only using its encoder module with causal masking.

These architectures can be initialized randomly or using the weights from a pre-trained model trained on general data using unsupervised learning objective such as T5 (Raffel et al., 2020). With the cross entropy loss,

$$L(\hat{y}, y) = \sum_{i=1}^{m} y_i \log \hat{y}_i \tag{3}$$

where $\hat{y}_i$ refers to the conditonal probability placed on the true label y_i by the model, these models are optimized through stochastic gradient descent through some kind of adaptive

learning rate optimizer such as Adam (Kingma and Ba, 2014). For Transformers, learning rate scheduler such as warm-up might be necessary for convergence when training a larger network (Popel and Bojar, 2018; Liu et al., 2020b). Regularizations such as weight-decay, dropout (Srivastava et al., 2014), or label-smoothing (Müller et al., 2019) are often employed but they are not necessary conditions for the convergence of these models.

2.2.3 Transformer and Positional Embeddings

In order to model the probabilities of sequences, the deep learning architecture must be sensitive to the positions of the tokens by representing them differently. LSTM accomplishes this by ingesting the tokens of a sequence sequentially one at a time and maintaining a memory vector to summarize the past tokens. However, Transformer and CNN are both feedforward networks attending to all the positions in parallel unlike LSTM. Due to the networks being solely feed-forward without any recurrence, they are inherently unaware of the positional information. The solution is to provide the model with an encoding that can elucidate the positional information.

For Transfomer, the method adopted in the original paper (Vaswani et al., 2017) used the absolute sinusoidal positional embedding that gets added to the word embedding before being processed by the Transformer layers. Given a model with the model dimension of d_{model}, the proposed PE encodes generated an embedding vector of size d_{model},

$$\begin{aligned} \text{PE}(pos, 2i) &= \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right) \\ \text{PE}(pos, 2i+1) &= \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right), i = 1, 2, ...d_{model} \end{aligned} \tag{4}$$

This PE is *absolute* as each position is represented as a unique vector. However, the model can still recover the relative positions from the absolute positions due them being composed of sine and cosine functions with different frequencies. Also, the sinusoidal functions allow the model to more easily generalize to unseen positions as the PE values are bounded and change predictably. There exists other position encoding methods that rely on leveraging only the relative position information (Dai et al., 2019). These methods add the relative position encodings not as an additive embedding to the word embedding but directly modify the attention matrix by providing a bias or embedding vector for each position offset $i - j$. We do not consider these relative positional encoding variants in this work.

2.3 Compositional Geneneralization in Deep Neural Networks

Various works have noted the lack of compositional generalization capability in deep learning models introduced in the previous section. We cover some of the most notable works regarding this flaw in Section 2.3.1. However, other works have shown conflicting results by noting the emergence of systematicity in the same models upon a change in the data distribution. This emergence includes the large language models. We present these experiments in detail along with the emergence of in-context learning in large language models in Section 2.3.2.

2.3.1 Lack of Compositionality in DNNs

We focus on the reported results on unimodal language data for the lack of compositionality in deep learning models [1]. Table 1 and 2 summarizes the results (see Appendix A for details on the datasets and their compositional splits).

Text-to-Text using Synthetic Data. The above mentioned sequence to sequence models were tested on synthetic text-to-text translation datasets of relatively small to medium size (Lake and Baroni, 2017; Hupkes et al., 2020; Liška et al., 2018; Korrel et al., 2019). Notable datasets are SCAN (Lake and Baroni, 2017; Loula et al., 2018; Dessì and Baroni, 2019) and PCFG (Hupkes et al., 2020) which are both constructed using context free grammar rules [2]. They were able to generalize successfully on the test set when the train and test were split through random sampling (i.e. i.i.d condition). However, for the compositional splits of the same datasets, all models generalize considerably poorly though the performance of the models varies from dataset to dataset (see Table 1). Fine-tuning a large pre-trained model such as T5 (Raffel et al., 2020), which became the standard approach for virtually all NLP tasks, shows some improvement over the base Transformer model on some of the splits but still fails to generalize successfully on every split. For instance, the 11 billion parameter version of T5 shows a mere ∼26% accuracy on the easiest of the MCD splits of SCAN (Conklin et al., 2021). One interesting finding is that fine-tuning a larger pre-trained model shows a worse performance on some of the compositional splits which goes contrary to the conventional wisdom of fine-tuning for NLP.

Semantic Parsing to Capture Natural Language Variation. Following the works using synthetically-generated artificial language datasets to test compositional generalization, many works have noted the lack of linguistic variation in these datasets which makes them not representative of natural language (Kim and Linzen, 2020; Keysers et al., 2020; Shaw et al., 2021). They instead proposed to use semantic parsing tasks to measure compositional generalization capability of DNNs which can better capture the natural language variation. Semantic parsing tasks require mapping of a natural language instruction into a logical form (e.g. SQL query) which can be used to query from the database. Notable datasets are CFQ (Keysers et al., 2020) and COGS (Kim and Linzen, 2020) which are generated automatically using grammar rules but more closely resemble the real natural language distribution. The compositional splits on CFQ and COGS are generated based on MCD and on selected linguistic generalization phenomena respectively. The results are similar to the synthetic datasets where the models are able to easily generalize for the i.i.d setting but not for the compositional splits, see Table 2.

The apparent lack of systematicity in DNNs has led to many researchers arguing for the need of a stronger inductive prior for compositionality in deep learning models to better guide thems towards a compositional solution. On the other hand, some argued for the need for a

[1] We do not include works in grounded language setting. The notable datasets are CLEVR (Johnson et al., 2016), SQOOP (Bahdanau et al., 2019), and gSCAN (Russin et al., 2019) and the results similarly show a lack of compositional generalization capability in DNNs.

[2] Some works have also proposed arithmetic mathematical and logical reasoning tasks to study their ability to learn the underlying hierarchical compositional structure for generalization (Hupkes and Zuidema, 2018; Saxton et al., 2019).

	SCAN						PCFG		
	r	l	j	mcd1	mcd2	mcd3	**r**	p	s
LSTM	99[1]	2.5[1]	1.2[1]	4.7[3]	7.3[3]	1.8[3]	79[4]	30[4]	53[4]
CNN	99[2]	0.0[2]	69.2[2]	-	-	-	85[4]	31[4]	56[4]
Transformer	99[5]	0.0[5]	1.0[5]	0.4[3]	1.8[3]	0.5 [3]	92[4]	50[4]	72[4]
T5-base	99[3]	14.4[3]	99.5[3]	26.2[3]	7.9[3]	12.1[3]	-	-	-
T5-11b	99[3]	2.0[3]	98.3[3]	7.9[3]	2.4[3]	16.8[3]	-	-	-

Table 1: Standard sequence to sequence models' performance on notable synthetic text-to-text datasets using exact-match accuracy. The first row shows the dataset name and the second row specifies the name of the compositional split. The split "r" refers to the random splitting of the dataset. Others refer to a particular compositional split of the corresponding dataset. For example, "SCAN-mcd1" refers to one of the compositional splits of SCAN based on MCD. References: [1] (Lake and Baroni, 2017), [2] (Dessì and Baroni, 2019), [3] (Furrer et al., 2021), [4] (Hupkes et al., 2020), and [5] (Ontañón et al., 2022)

	COGS		CFG			
	r	g	**r**	mcd1	mcd2	mcd3
LSTM	99[1]	16[1]	97[2]	28.9[3]	5.0[3]	10.8[3]
Transformer	96[1]	35[1]	98[2]	34.9[3]	8.2[3]	10.6[3]
T5-base	-	-	-	57.6[3]	19.5[3]	16.6[3]
T5-11b	-	-	-	61.4[3]	31.3[3]	31.2[3]

Table 2: Standard sequence to sequence models' performance on notable semantic parsing tasks using accuracy. The first row shows the dataset name followed by the name of the compositional split. The split "r" refers to the random splitting of the dataset. Others refer to a particular compositional split of the corresponding dataset. For example, "CFQ-mcd1" refers to one of the compositional splits of the dataset CFQ based on MCD. References: [1] (Kim and Linzen, 2020), [2] (Keysers et al., 2020), and [3] (Furrer et al., 2021)

hybrid neuro-symbolic approach combining the strength of the symbolic AI systems capable of compositionality and the strength of representation learning capability of deep learning models (see Section 3.1.2 for more detail).

2.3.2 Emergence of Systematicity by the Property of Data Distribution

However, some works have reported on an emergence of systematicity for the modern deep learning models without any modification (Patel et al., 2022; Prabhu Prakash, 2020; Hill et al., 2020). In these works, only the property of the underlying data distribution used for the training were altered while maintaining their systematic split to the test set.

Emergence of Compositionality from the Compositional Splits. Patel et al. (2022) extended the number of primitives in SCAN for the compositional split where only a lone primitive mapping "*jump* $\rightarrow$ *JUMP*" seen during training is tested on every possible use of the primitive in a compound (see SCAN-j in Table 1). For example, the new primitive

"$swim \rightarrow SWIM$" was added to the dataset along with their uses in a compound such as "$run\ and\ swim \rightarrow RUN\ SWIM$". The original split only had three other primitives which are seen in full in a compound. It was found that they were able to generalize perfectly for the new primitive upon seeing more than a hundred primitives in the training set compared to the three primitives in the original setting.

Hill et al. (2020) also reported with the similar finding for a situated agent manipulating a simulated environment upon a natural language instruction such as lifting or locating an object. The zero-shot generalization capability of the agent for the action verbs depended on the number of the unique objects experienced in conjunction with the action during training. Only when the agent learned to perform a certain action for a diverse set of objects, the skill generalized to novel objects seen in different contexts.

Hence, systematicity can be discovered by the deep learning models when the data distribution contains a certain level variability that can properly reveal its the invariant structure. These findings mirrors that of human generalization pattern where more diverse contexts allow for better and more robust generalization (Gómez, 2002; Johns et al., 2016; Norman et al., 2022).

Emergence of Structural Knowledge in Language Models. The use of the compositional splits provides direct evidence for the model's compositional generalization capability as we are performing a controlled experiment (see Section 2.1.3). However, it is also of our interest to study the knowledge of DNNs trained on a real-world data. As mentioned above, this is difficult to do and we can only gain an indirect evidence from evaluation as their training data cannot be strictly controlled for.

One such a model of our interest is the language model (LM) which is trained on a large collection of natural language corpora such as Wikipedia using the token prediction objective (Equation 2). Several works have studied whether the LMs generalize using heuristics or through the use of structured grammatical knowledge (Linzen et al., 2016; Chowdhury and Zamparelli, 2018; Gulordava et al., 2018; Lakretz et al., 2019). The results show an emergence of generalization capability on some basic linguistic grammatical operations such as the subject-verb number agreement in different languages even when the distance between the subject and the verb were made more distant and the semantic cues were controlled by making the sentences nonsensical (e.g. "Colorless green *sleeps* restlessly") (Gulordava et al., 2018; Lakretz et al., 2019).

More recently, emergent capability of the **large language models** (LLMs) (Brown et al., 2020; Chowdhery et al., 2022) in exhibiting **in-context learning** (Garg et al., 2022) was found where it can learn by from a few demonstrations provided in its context without any changes to the weights. Their in-context learning ability can be further boosted through *augmented prompting* methods (Wei et al., 2022b; Zhou et al., 2023; Wei et al., 2022a) such as the chain-of-thought prompting (Wei et al., 2022b) which provides step-by-step demonstrative examples by breaking down how the overall task can be solved by solving successive sub-problems. Through the few-shot or augmented prompting, LLMs have shown to be capable of solving many mathematical reasonings, linguistic generalization, and compositional generalization tasks (Wei et al., 2022b; Zhou et al., 2023; Lampinen et al., 2022; Wei et al.,

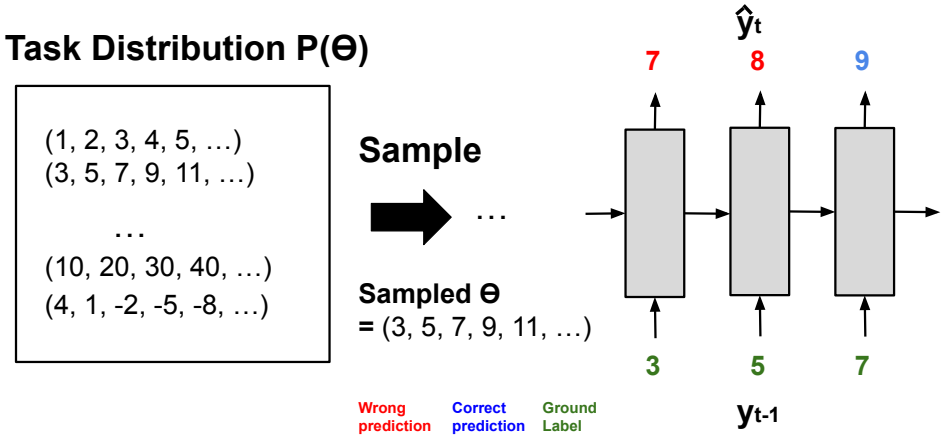


Figure 2: An illustration of meta-memory learning for learning to learn sequences of numbers of a linear function. We can define a distribution of linear sequences and sample the sequences to feed the model one value at a time. The model receives the past time-step's true ground input y_{t-1} which the model integrates into its memory for the prediction in the next time-step y_t. Since the model does not know which sequence is given at each episode, it must learn and improve upon experience.

2022a; Anil et al., 2022).

2.4 Memory-based Meta Learning

Meta-learning is a machine learning framework that aims to learn the inductive priors directly from data rather than designing them into the model (Bengio et al., 1991; Schmidhuber et al., 1996; Thrun and Pratt, 1998). This approach trains a model on a distribution of tasks rather than a single task. This way, the model *learns to learn* a general-purpose strategy that can be used to succeed on all the tasks by learning the structure of the task distribution.

Any meta-learning system can be conceptualized as consisting of three separate systems: a **learner**, **supervisory system**, and **update system**. Given a task, the learner performs the task and the supervisor provides how well the learner had performed Given the supervisory signal, the update algorithm is used to improve the learner such that the learner can better perform the task in the next trial. For illustration, imagine us learning to write a computer program for a particular problem, we, as a learner, write the program and check whether the code is sound by compiling the code (i.e. the supervisory system). If the code does not compile, we use this supervisory signal to improve our code where the update algorithm is executed somewhere in our brain. Note that the update system should not be confused with the process of improving the learner to better learn (i.e. the *meta*-update system).

Most of the meta-learning approaches assume that the supervisory system is given in the form of true ground labels of a dataset or a pre-trained system that can provide the signal upon request (e.g. auto-regressive language model). However, they differ on how the learner and the update system are defined. In the memory-based meta-learning (MBML) approach, rather than the update algorithm being designed by humans, it is learned in the dynamics

of the learner from data alone. This is possible because the memory-based meta-learner uses a memory to keep track of the past observations and the past supervisory signals and incorporate the new supervisory signal in the memory to improve itself. Hence, it provides a way to learn a *general* learning algorithm that can improve from experience for the task distribution with minimal inductive biases.

We provide a more detailed setup of MBML in Section 2.4.1. Then we provide some general considerations and perspectives related to MBML that are important for understanding our work in Section 2.4.2.

2.4.1 Problem Setup

Memory-based meta learning is a type of meta-learning which can train models that can learn from experience and improve in a data efficient-manner (Hochreiter et al., 2001; Duan et al., 2017; Ortega et al., 2019) (see Figure 2 for illustration). This can be applied when learning over sequences $\tau = \{x_1, x_2, ..., x_N\} \in \mathcal{V}^N$ where $\mathcal{V}$ is a finite alphabet of observations. Each task can be considered a particular probability distribution P over sequences with the index θ : $P(\tau|\theta)$. Then defining a task distribution can be thought of as defining the prior distribution $P(\theta)$. Then given a parametric model f_ϕ that can model the probabilities of sequences using the policy $\pi_\phi(\tau)$ (i.e. $f_\phi(\tau)$ maps to probabilities π), the goal of MBML is to minimize the expected loss on the task distribution

$$\mathrm{E}[l] = \mathrm{E}_{\theta \sim P(\theta)}[\sum_{\tau} P(\tau|\theta) l(\tau, \pi)] = \sum_{\theta} P(\theta) \sum_{\tau} P(\tau|\theta) l(\tau, \pi) \tag{5}$$

where $l : \mathcal{X} \times \mathcal{X} \to \mathcal{R}$ is the loss function. Meta learning aims to find the minimum of the expected loss through *Monte-Carlo approximation*

$$\mathrm{E}[l] \approx \frac{1}{N} \sum_{n=1}^{N} l(\tau^{(n)}; \pi(\tau^{(n)})) \tag{6}$$

where trajectory samples $\{\tau^{(n)}\}_{n=1}^{N}$ are drawn i.i.d by

$$\tau^{(n)} \sim P(\tau|\theta^{(n)}), \theta^{(n)} \sim P(\theta) \tag{7}$$

where the superscript refers to the index of the samples. For the MBML models, the choice of f_ϕ is such that it can model the conditional probabilities $f_\phi(x_{<i}) = \pi(x_i|x_{<i})$. These models instantiate this form by the memory vector m_{t-1}, keeping the sufficient statistics of the past: $f(x_{<i}) = f(x_{t-1}, m_{t-1})$. Then we can expand the Monte Carlo approximation of Equation 6,

$$\mathrm{E}[l] \approx \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T} l(x_t^{(n)}; f(x_{t-1}^{(n)}, m_{t-1}^{(n)})) \tag{8}$$

The pseudo-code used to train a meta-memory model using MBML is listed as Algorithm 1. One iteration of Monte Carlo approximation of the expected loss is known as a **meta-episode** and it consists of samping N **trajectories**. Each trajectory is rolled out until the maximum length T.

Algorithm 1 Memory-based Meta Learning for Sequential Predition Task

Given task distribution $P(\theta)$, batch-size N, maximum roll-out length T, loss function l, meta-learner f
while not converged **do**
    $L \leftarrow 0$
    for $n = 1$ to N **do**
        Sample task $\theta^{(n)}$ from $P(\theta)$
        Initialize starting memory state $m_0^{(n)}$
        for $t = 1$ to T **do**
            Sample observation $x_t^{(n)}$ from $P(x_t|x_{<t}, \theta)$
            $L = L + l(x_t; f(x_{t-1}^{(n)}, m_{t-1}^{(n)}))$ (accumulate loss)
            Save memory $m_t^{(n)}$
    minimizationStep(f, L)

The optimal minimizer $\pi^*(x_t|x_{<t})$ of the expected loss is the Bayesian model that computes the posterior predictive probability over all possible tasks

$$\pi^*(x_t|x_{<t}) = \sum_{\theta} P(\theta|x_{<t})P(x_t|\theta, x_{<t}) \tag{9}$$

from past data (Ortega et al., 2019). Hence, a memory-based meta learning model trained through minimizing Eqn 6 learns to implicitly compute the Bayesian inference update automatically. Since the Monte Carlo sampling hides the identity of task as it marginalizes over the task distribution, the model needs to use the past experience to improve the loss by building an internal model of the world in its memory. Hence, given the task distribution with some underlying structure, the model can learn to extract this structure from data to improve from experience (i.e. in-context learning). Of course, whether we can actually find the minimizer f^* depends on the type of functions f_ϕ can model in theory and how difficult the optimization is in finding the minimizer for the given model in practice.

2.4.2 Important Considerations regarding Meta-memory learning

LSTM and Transformer are Meta-Learners. We note that LSTM and Transformer satisfy the conditions to be memory-based meta-learners when working with natural language data (or any type of sequence data). We have observed that a memory-based learner f modelling the conditional probabilities of the observations in trajectory $\pi(x_t|x_{<t}) = f(x_{t-1}, m_{t-1})$ can be a meta-memory learner given the ground observation from the past time step x_{t-1} using its memory m_{t-1}. When trained on a distribution of tasks minimizing the expected loss, it can be a meta-learner.

The sequence to sequence model and language models using LSTM or Transformer are trained unidirectinoally using teacher-forcing where the ground truth past tokens are provided in predicting the next token. Once, sufficiently trained, these models can also leverage their own prediction to approximate the supervisory signal in an autoregressive manner.

Hence, the models such as LSTM and Transformers are *potential meta-learners* depending on the data distribution used for their training. Thus, once trained, these models have a potential to exhibit in-context learning.

Informative Trajectories and Size of the Task Distribution. Training on a diverse sequence prediction task composed of sequences with latent patterns that the model can utilize for improvement is equivalent to MBML. This is the case for language modelling as the model can use the information contained in the previous words to accurately predict the next word. Furthermore, due to the "burstiness" of words (Altmann et al., 2009) in natural language, the occurrence of some words are highly contextual which makes the natural language sequence to be highly informative.

However, even with informative trajectories, the ability to find the general-purpose strategy also depends on the size of the task distribution. For the sufficiently capable model, memorizing the entire task distribution might be a easier solution than finding a general-purpose strategy. This was illustrated in the experiments of Chan et al. (2022) where a hand-written character classification task was framed as a prediction in the last position of the sequence containing the image-label pairs needed for the prediction . The sequence contained other distractor pairs. Using Transformer, the model did not know any signs of in-context learning for 100 classes. However, when the unique number of classes from 100 to 12800, this led the in-context learning to emerge.

Spontaneous Meta Learning in Online Environment. For online learning algorithm training on long history of past states using a model with a bounded capacity, meta-learning can spontaneously arise as the model will effectively perform batch updates on the trajectories from different tasks (Ortega et al., 2019). This is because some memory state m^* of an online-meta learner is bound to repeat even when the true latent parameter of the data distribution does not as the model with limited capacity can only approximate the true data generating parameters. Hence, when the context window is sufficiently long, this leads to the trajectories separated by the repeating memory state m^* to be updated *in batches* approximating the Monte-Carlo approximation of the expected loss of Equation 6.

3 Related Works

Our work lies at the intersection of two areas in artificial intelligence: compositional generalization and meta-learning. We leverage and build upon numerous prior studies conducted in both these fields. We first lay out the related prior works on compositional generalization in Section 3.1. The topic of compositional generalization in AI has been discussed for decades and understanding this historical context allows a better understanding the significance of the approach we take in our work. We will begin by providing a concise summary of this discussion, followed by an overview of the existing literature on enhancing compositionality within deep learning models. Next, a general overview of the field of meta-learning is presented in Section 3.2 focusing on their applications in the field of NLP. Lastly, we review the prior works on in-context learning of both emergent and designed origin.

3.1 Compositional Generalization

The central topic of the thesis is compositional generalization where its importance in building intelligent machines has been noted for decades. In fact, the idea was first presented and developed through a series of debates on the limits in the field of *connectionism* which developed into the field of deep learning today. To provide historical context emphasizing the significance of compositional generalization, we present the key arguments driving this debate in Section 3.1.1. This allows us to better understand the various works that have proposed distinct inductive biases to address the issue of compositionality deficiency in DNNs. These works will be elucidated in Section 3.1.2.

3.1.1 Debate on Compositionality in Deep Learning Models

The idea of the *emergence of rule-following behaviour* in neural networks is not new. This idea was made concrete during the classical-connectionism debate which cemented the importance of compositional generalization in building AI. The approach of classical or symbolic AI attempts to build AI as an algebraic rule system manipulating atomic and complex symbols (Fodor and Pylyshyn, 1988). Symbolic-AI was a dominant approach in the field of AI in the 80's when the *connectionist* approach (i.e. deep learning) based on distributed representation learning started to gain more attention. This clash led to a series of debates on the limits of the connectionist approach in being able to operate as a rule system (Fodor and Pylyshyn, 1988; Smolensky, 1988; Marcus, 2001; McClelland et al., 2010). This was important as the principle of human cognition was considered a rule-based system capable of manipulating symbols.

The main criticism laid out by the classicists was on the lack of compositionality in the connectionist architectures, unable to exhibit systematicity and productivity, due to the nature of the distributed representations. The connectionists agreed on the importance of compositionality, but they argued that the rule-following behaviours are *emergent*, which can be learned by the neural networks rather than being hard-coded into the networks. This was demonstrated in small-scale experiments (Elman, 1990). These criticisms also led to work on developing new connectionist architectures that are more sensitive in learning structural knowledge (Smolensky, 1990).

The main arguments from the debate survives even today in the modern "deep learning era" in search of compositionality for DNNs. Some works proposed the hybrid neuro-symbolic approach as a solution while others have attempted to provide better inductive biases for DNNs to behave more like the symbolic systems. In this work, we take the other side of debate by studying the emergence of compositionality of DNNs with minimal inductive bias.

3.1.2 Inductive Priors for Compositional Generalization

With the introduction of the new benchmarks that showed the lack of compositionality in modern deep learning models (see Section 2.3.1), many works have proposed different ways of providing the inductive biases for compositionality. Here we present some of the notable works for unimodal text data by categorizing by the type of employed inductive bias.

New Deep Learning Architectures. One line of work attempts to instill compositionality into the deep learning models as the structural inductive bias in how the information is represented and processed by the models. The main objective behind these works is to aid in building disentangled representations that can separate the meaning from form which is the main principle of compositionality.

Korrel et al. (2019) introduced additional RNN between the encoder and decoder to better regulate the attention mechanism to make the alignment process the inputs and the outputs more sharp. Simiarly, Akyurek and Andreas (2021) introduced a lexicon to sequence to sequence model to better disentangle the lexical phenomena from the syntatic ones. Li et al. (2019) and Russin et al. (2019) proposed to model the syntax and semantics separately. They trained two encoders separately representing for syntax and semantics where the syntactical representations were used to select which semantic representations will be utilized for predictions. Gordon et al. (2020) hypothesized that compositionality of language is a form of group-equivariance and trained permutation-equivariant sequence to sequence models. Bergen et al. (2021) modified the Transformer network with relation information and a special attention mechanism to better learn the latent structure of language.

These works led to a great improvement in compositional generalization in synthetic datasets such as SCAN. However, some of these models were shown worse performances on larger datasets that better captures the large variations in natural language (Furrer et al., 2021). Furthermore, it is noteworthy to mention that none of these works have evaluated their novel architectures on prominent NLP benchmarks. These considerations raise concerns about the scalability of these models in effectively capturing and modeling larger-scale natural language data.

Searching among Available Architectures. Some works have focused on evaluating the generalization difference between the available variants of the popular deep learning architectures. Ontañón et al. (2022) and Csordás et al. (2021) conducted various studies on Transformers and found that the use of relative positional encoding (Dai et al., 2019), the choice of positional encoding initialization, sharing of layer weights (Dehghani et al., 2019), and an additional copying mechanism from the encoder to decoder can lead to better generalization performance. Furrer et al. (2021) found that using a pretrained sequence to sequence model for finetuning can also lead to some improvement which is the standard practice in

most NLP tasks. These positive results show that the base deep learning architectures could be further improved with better inductive biases.

Additional Supervisory Signal. Another line of work attempts to provide an additional supervisory signal for the model to more easily arrive at the desired parameter space with compositionality. Hupkes et al. (2019) provided a guidance signal on the attention of an RNN based sequence to sequence model to be able to perfectly generalize on a look-up table task (Liška et al., 2018). Jiang and Bansal (2021) provided additional hand designed sequences that can elucidate the structural boundaries of action sequences in SCAN as the additional targets for the Transformer to predict. They found that this additional prediction task can lead an improved generalization performance. Finally, for various compositional generalization datasets, Herzig et al. (2021) showed that using a different target representation that can be easier for the pretrained models to output can lead to an improvement in the generalization performance. These works serve as additional evidence that modern deep learning models are indeed capable of learning the function classes with compositionality.

Data Augmentation. The next line of work is data augmentation where the data distribution is augmented with synthetic examples created by recombining the training examples. Andreas (2020) proposed a simple algorithm that can use the fragments that occur in multiple environments in a similar manner to generate synthetic examples for SCAN and showed that this simple approach can lead to a drastic improvement in performance. Akyürek et al. (2021) extended this idea further by training an RNN that can generate synthetic examples from the fragments through recombining. Guo et al. (2020b) used the iterated back-translation (Edunov et al., 2018) to augment the training dataset for improvement. However, these works do not maintain the systematic split between the train and test split as some of the generated synthetic data come out to be the actual examples from the test distribution.

Neuro-symbolic Apporoach. Others proposed the hybrid neuro-symbolic approaches that can explicitly perform the compositional computation by design. Nye et al. (2020) took the neural program synthesis approach by training an RNN encoder to generate a set of grammar rules that can be used to deterministically translate the SCAN commands for prediction. Liu et al. (2020c) and Liu et al. (2021) used Tree-LSTM (Tai et al., 2015) to directly model the symbolic functions with variable slots to solve compositionality. These were trained in an end-to-end manner through reinforcement learning (RL) using curriculum learning (Bengio et al., 2009). Guo et al. (2020a) argued that the sequence to sequence problem can be framed as a decoding problem of partially order sets (poset) which can better capture the lexical permutation invariance structure of language (e.g. A and B is equivalent to B and A). Then using several individually trained modules, the poset predicted for a given sequence which is then converted into an output sequence. Chen et al. (2020) introduced a symbolic stack machine that can be used to generate sequences. These neuro-symbolic approaches have shown near-perfect results on the synthetic datasets such as SCAN. However, it is unclear whether it would be possible to scale these models to properly capture the variation and exceptions that exists in natural language.

Unlike the presented works in this section, we do not attempt to design an inductive bias for compositionality. Rather, we attempt to learn the inductive bias directly from data through

MBML using the established deep learning architectures.

3.2 Meta-Learning

Next relevant field for our thesis is meta-learning. Meta-learning endeavors to enable machine learning models to learn how to learn by exposing them to various tasks that possess similar structures but differ in specific details. We provide a brief overview on the different learning to learn approaches in Section 3.2.1. Section 3.2.2 follows by covering how meta-learning is applied in the field of NLP along with a general discussion on why the MBML approach is not popular in the field. The section concludes with an overview of the in-context learning approaches in machine learning including the emergence in LLMs (Section 3.2.3).

3.2.1 General Overview of Meta-learning

The capability to learn the update algorithm from data alone makes the meta-memory learning approach to be more general than the other meta-learning approaches such as the *metric-based* or *optimization-based* approaches which rely on a human-designed update system.

Few-shot Learning. Before explaining how the other two approaches differ from MBML, we introduce the concept of *few-shot learning* as these approaches can be more easily understood by slightly altering the meta-learning problem setup introduced in Section 2.4.1. In a few-shot learning problem, a model needs to improve from the **support set** S of consisting of k examples. For the classification problems, the support set consists of feature-label pairs $S = \{(x_i, y_i)\}_{i=1}^{K}$. Then it is tested on its performance on the **query set** Q consisting $|Q|$ novel examples without labels. This is called the **k-shot learning** problem as the model is given k relevant support examples for generalization.

In MBML, the model needs to use the past n observations or examples to successfully predict the $(n+1)^{\text{th}}$ observation for all $n \in 1, 2, ...|\tau| - 1$. Hence, the memory-based meta-learner learning on a trajectory τ of length $|\tau|$ implicitly solves $(|\tau| - 1)$ few-shot learning problems with $|Q| = 1$.

Metric-based Meta-learning. The metric-based meta-learning approach for few-shot learning trains some metric function $f(x_1, x_2)$ which can compute the distance between the two examples x_1 and x_2. This metric function can be used with a non-parametric inference model $P(y|S)$ of the form

$$P(y_s|S, x_q) = \frac{\exp(f(x_s, x_q))}{\sum_{x_r \in S} \exp(f(x_r, y))} \tag{10}$$

where the support set S is assumed to consist of feature-label pairs $\{(x_i, y_i)\}_{i=1}^{K}$ and x_q is the query example. Hence, the update system is pre-defined in the form of a distance-base classification algorithm. Some notable metric models are Siamese (Koch et al., 2015), Matching (Vinyals et al., 2016), and Protytypical network (Snell et al., 2017).

Optimization-based Meta-learning. The optimization-based approach instead uses the intuition that the update system can be naturally framed as an optimization problem of the

learner through some loss function l. One of the most popular work is MAML (Finn et al., 2017) which uses the back-propagation and gradient descent as the update algorithm to improve a differentiable learner f with parameters θ. Given a support set $S = \{(x_i, y_i)\}_{i=1}^{K}$, MAML improves f on the support set by updating the parameters using gradient descent. The updated learner with the new parameters θ^* then can be used for testing on a query example x_q,

$$P(y|x_q, \theta^*), \theta^* \leftarrow \theta - \alpha \sum_{i=1}^{K} l(y_i, f(x_i); \theta) \tag{11}$$

where α is the step-size and l is the loss function. Hence, MAML can be interpreted as an approach that meta-learns a good initialization that can be quickly adapted with a few gradient updates using transfer learning. There exists other variants of MAML such as REPTILE (Nichol et al., 2018) that differ on the details of how the meta-update is made. Other works attempt to learn the optimization algorithm that can substitute the gradient descent (Ravi and Larochelle, 2016; Li et al., 2017; Li and Malik, 2017; Andrychowicz et al., 2016) or learn the loss function for better optimization(Kirsch et al., 2020).

3.2.2 Meta-Learning for NLP

Constructing a task distribution to learning generally transferable language representations in NLP is challenging. The difficulty arises from the uncertainty of identifying beforehand which examples can effectively serve as suitable support examples. Hence, the focus of meta-learning approaches in the field has primarily been on addressing low-resource transfer learning problems, rather than being a central paradigm like the unsupervised pre-training. Especially, the use of MAML variants has been widely applied for the low-resource settings on transferring cross-lingually for machine translation (Gu et al., 2018), question-answering (Nooralahzadeh et al., 2020), and automatic speech recognition (ASR) (Hsu et al., 2019; Winata et al., 2020) or for domain adaption in dialogue generation (Qian and Yu, 2019), and speaker-adaption in ASR (Klejch et al., 2019).

Under-utilization of MBML in NLP. MBML has been underutilized in NLP even though it can learn its own update system from data and this makes this approach more suitable for its application with natural language data. The underlying latent structure of natural language data is complex, diverse, and heavily context-sensitive where not all the support examples are equally relevant for each task. The under-utilization can be attributed to the computational overhead of a meta-memory learner. For most NLP tasks involve processing long sequences, the meta-memory learner encounters the challenge of conditioning generation on a large number of representations which can be computationally expensive. On the other hand, MAML distills the knowledge into the model parameters and do not require more computation after the adaptation. Although we encounter a similar challenge in our work, it is not the primary focus. Consequently, we operate within the configurations that are feasible for our research.

Meta-learning for Compositional Generalization. Before moving on the next section, we review the few works that pursued the meta-learning approach for compositional generalization which is the line of study of our work. Lake (2019) proposed a memory-based

meta-learning sequence to sequence LSTM in SCAN by training on all permutations of the primitive mappings. During the testing phase, the model was supplied with the initial primitive mappings for generalization and it demonstrated a perfect generalization performance. However, this work relies on the knowledge of the grammar to generate the task distribution. Conklin et al. (2021) used MAML as a regularizer in the loss function of supervised learning and showed improvement in generalization on several MCD splits of SCAN and COGS. However, they also relied on hand-designed methods to design the task distribution. In our work, we use minimal inductive bias to generate the task distribution by relying on the spontaneously arising k-shot learning problems during an online learning of the dataset.

3.2.3 In-context Learning in DNNs

In-context Learning in LSTM. Many works have studied the in-context learning capability of LSTM and Transformer on various complex machine learning tasks. LSTM was most widely studied in the meta RL environments and was shown to behave like a model-based reinforcement algorithm, successfully extracting the structure of the learning environment (Duan et al., 2017; Wang et al., 2017). The meta-RL approach provides the model with past rewards as the supervisory signal. The meta-RL approach has to solve a sequential decision-making problem which is different from the sequential-prediction problem for NLP problems explored in this work. This is because the environment changes with the change of the agent's policy. Many works followed and proposed new deep learning architectures that can improve the limited context-window of LSTM for meta-RL using external memory (Wayne et al., 2018; Ritter et al., 2018, 2021; Fortunato et al., 2019).

As discussed above, MBML has not been popular with natural language data. Still, some works have shown the capability of the LSTM-based models in solving image-classification few-shot learning problems where the model needs arbitrarily bind the images and the labels for prediction (Santoro et al., 2016; Mishra et al., 2018).

In-context Learning in Transformer. Transformer was shown to be capable of in-context learning for the linear function class (Garg et al., 2022), image class categorization (Kirsch et al., 2022), and RL environments (Melo, 2021). Of course, the emergent LLMs have shown that Transformers can be a general in-context learner for a variety of natural language, mathematical reasoning, and systematic generalization tasks (Brown et al., 2020; Wei et al., 2022a).

The in-context learning capability of LLMs are *emergent* as it was never trained for a general ability to learn from examples. Also, the examples that are provided to LLMs are separated by a special delimiter symbol. The resulting sequences LLMs are able to in-context learn from are mostly near zero-probability sequences in their training distribution. Thus, LLMs are able to *generally* apply in-context learning even for the OOD sequences. Different works have attributed this emergence to the nature of natural language data distribution exhibiting burstiness of words and label multiplicity (Chan et al., 2022) (discussed in Section 2.4.2) or having to accurately predict the latent topic of documents for successful token prediction (Xie et al., 2022).

Difference between LSTM and Transformer. Finally, some works have found that

only Transformers and not LSTM are capable of in-context learning for more complex environments in their experiments (Santoro et al., 2016; Chan et al., 2022; Kirsch et al., 2022). LSTM suffers from its inability to attend to all the past observations with clarity due to its inherent sequential binding nature of information as well as its forget-gate which continuously discard information at each time-step. However, one recent work has demonstrated that LSTM can be made an in-context learner (Xie et al., 2022). This suggests that both are capable, but it is easier to optimize Transformer than LSTM to find the in-context learning solution.

We separate ourselves from prior work by learning to do *in-context learning on sequences* where the lengths of the inputs and labels can arbitrarily vary. Also, we use the in-context learning model to study its relation with the ability to compositionally generalize.

4 Memory-based Meta-Learning for Compositional Generalization

Our objective is to train an in-context learning model on sequence to sequence data. We hypothesized that learning to learn from examples implies the learning to extract the kind of latent structure that can be useful for compositional generalization. Hence, we propose a new MBML framework that can train a meta-learner with memory for in-context learning on a trajectory of sequences. We desire to do this using minimal inductive bias such that the proposed method has a potential to be generally applied to other sequence modelling tasks.

The main insight behind our work is that an online learning of the dataset can provide a strong signal for learning the latent structure of the data distribution. In a online learning trajectory, a model with memory observes each example in the dataset only once. Thus, it must maintain an organized memory of the past examples by successfully representing the rules governing the interpretation of the inputs to their corresponding outputs. The generalizability of this abstraction process is tested anew for each sequence in the trajectory.

Furthermore, different online trajectories of the dataset would present a different in-context learning task while the structure behind each trajectory remains invariant. Hence, by training a meta-learner with memory on this distribution of online learning trajectories, the model would be able to approximate the true generative structure of the dataset.

We can further increase the signal the discovery of the true parameters by making the in-weights learning impossible. This can be achieved by creating multiple versions of the dataset where their surface representations are shuffled. Then, when learning on online trajectories sampled from multiple versions of the same dataset, the model cannot memorize the answer for each example by ignoring what is in its context as multiple labels are assigned to each input. Hence, the only way the model can succeed would be through the use of previous examples in the trajectory for generalization.

We present the described approach more formally in the following sections (Section 4.1, 4.2, and 4.3) along with how this can be used for zero-shot evaluation (Section 4.4). The chapter concludes with a brief theoretical discussion on the limits of our proposed approach (Section 4.5).

4.1 Generating the Task Distribution by Permutation of Samples

We assume that we are given a dataset consisting of sequences $D = \{\tau^{(i)}\}_{i=1}^{N}$ with no particular ordering between the examples. Each sequence has a variable length m operating in a finite vocabulary $\mathcal{V}$: $\tau \in \mathcal{V}^m$. Specifically, we are working with sequence to sequence datasets consisting of pairs of input and output sequences $D = \{(\tau_x^{(i)}, \tau_y^{(i)})\}_{i=1}^{|D|}$. They can be made into a single sequence using special delimiters and only incurring loss for the output sequences.

As mentioned before, designing the task distribution is a challenge for natural language, thus we rely on the implicit generalization problems that arise in an online learning setting. Unlike the natural language data which has a natural ordering, sequence to sequence datasets

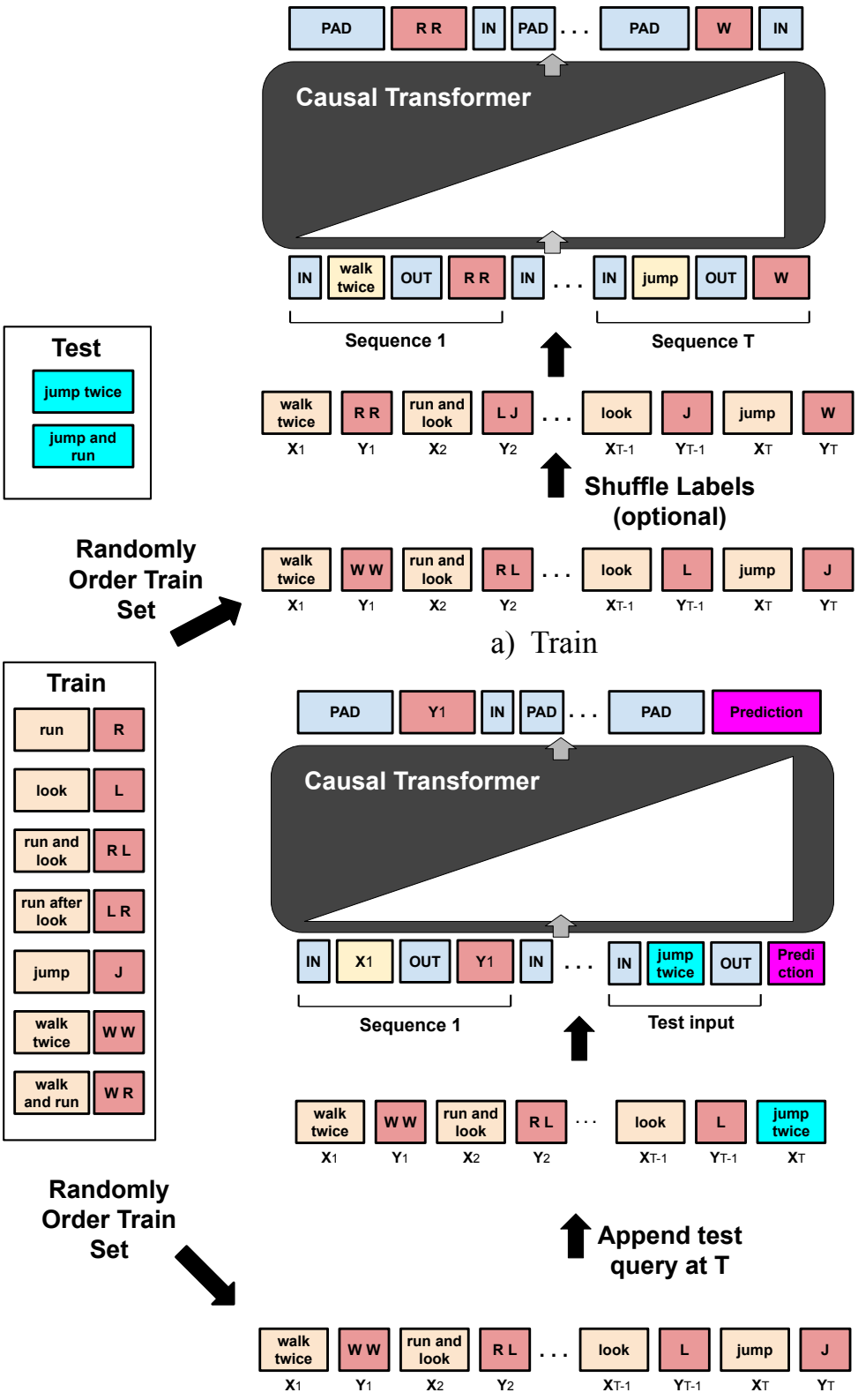


Figure 3: An illustration of the proposed meta-memory learning for compositional generalization.

do not have a predefined ordering. Hence, each task θ can be defined as a particular random ordering of the sequences (i.e. permutation of the dataset) in the dataset which results in a total of $N!$ different tasks. Each task in this distribution of tasks share the same latent structure as only the examples are simply rearranged. Hence, since the meta-learner learns depends on the shared structure of the task distribution, the structure underlying the dataset should be able to be discovered by the meta-learner when learning on this distribution.

Though the model can still exhibit in-weights learning by storing the answers in its weights as only the ordering of the datasets are altered, this is different from learning with a batch of sampled data points in parallel. A model that is sensitive to the past examples provided its context will be influenced by them and can learn to use them for prediction differently (i.e. in-context learning). In other words, given a meta-learner $f(x, m)$ with a limited capacity that cannot generate the same sufficient statistics vector m for differently ordered set of sequences, the different orderings essentially appear as different tasks for the learner. This is especially the case when the past examples are highly informative and the size of the task distribution is large for the model to easily memorize. Hence, in such a condition, learning to generalize by in-context learning might be a easier route for success then in-weights learning.

4.2 Generating the Task Distribution by Permutation of Labels

Another way, though not orthogonal, of generating a distribution of tassk is by defining different permutation functions of the vocabulary. We assume that the vocabulary $\mathcal{V}$ is an ordered set with an index function $\Delta : \mathcal{V} \to \mathbf{N}_{|\mathcal{V}|}$[3]. Such an indexing function is always defined to convert the text data to word embeddings.

Then, we can construct an alternate version of the dataset D by defining a permutation function $g : \mathbf{N}_{|\mathcal{V}|} \to \mathbf{N}_{|\mathcal{V}|}$ of the indexed vocabulary $\mathcal{V}_\Delta$ which rearranges the order of the vocabulary. For example, given a dataset $\{ab, cd\}$ with $\mathcal{V} = \{a, b, c, d\}$ with the ordering of $\{0, 1, 2, 3\}$ which can be used to represent the dataset as $\{01, 23\}$, we can create an alternate version of the same dataset by defining g as $g(0) = 1, g(1) = 2, g(2) = 0, g(3) = 3$. Then the dataset can be newly represented as $\{12, 03\}$. Both versions of the dataset have the same latent structure as only the integer mapping of the tokens were changed.

One can imagine doing this procedure for writings in English by simply swapping the order of the English alphabet. The underlying meaning of the writings would remain the same and only their surface representations would change. Under this framework, it is possible to generate the task distribution of size $|\mathcal{V}|!$ which is the number of possible permutations for a set of size $|\mathcal{V}|$. In this task distribution, each task is a particular permutation of $\mathcal{V}$. We can also control the size of the task distribution by limiting the number of allowed permutation functions or by increasing the size of the vocabulary. Generating the task distribution in this way prevents the possibility of in-weights learning as multiple labels are associated with each word. This means that the model *must* use the provided past examples for generalization.

There is another way to generate the task distribution while still allowing the possibility of in-weights learning. This can be done by extending the vocabulary $\mathcal{V}$ with another vocabulary

[3] $\mathbf{N}_n = \mathbf{N} - \{n + 1, n + 2, ..\}$

$\mathcal{W}$ that is disjoint ($\mathcal{V} \cap \mathcal{W} = \emptyset$). We can then define a bijection function (i.e. one-to-one) between $\mathcal{V}$ and $\mathcal{W}$ and being careful to only use one vocabulary at a time[4]. The number of tasks in the distribution of tasks can then be easily controlled by increasing or decreasing the number of new vocabularies. However, it is also computationally prohibitive as the overall vocabulary size grows linearly with respect to the number of tasks. Hence, we use permutation functions with a fixed vocabulary to generate the task distribution.

4.3 Implementation

In summary, each task distribution $\theta = P(\tau|\theta) = \prod_t P(x_t|x_{<t,\theta}, \theta)$ is a particular online ordering of sequences from the given dataset D possibly with a certain permutation function g. The sequences are concatenated using a special delimiter token which is not assumed to be included in the permutations of the vocabulary as it simply holds the boundary information. Since we are working with sequence to sequence datasets, the input and output sequences are concatenated using special delimiters "IN" and "OUT" as $[IN, x, OUT, y]$. The delimiter "IN" takes the role of the delimiter between different sequences and "OUT" takes the role of separating the input from the output. Then each trajectory τ of a task θ looks as follows: $[IN, g(x^{(0)}), OUT, g(y^{(0)}), IN, ..., g(x^{(|D|)}), OUT, g(y^{(|D|)})]$.

In practice, it is not possible or desirable to roll out the entire dataset for each sample trajectory when implementing the memory-based meta-learning algorithm described in Section 2.4.1. For example, LSTM is known to suffer from limited context window where at each step some information is forgotten through the forget gate, hence it cannot utilize all the past examples even if they are provided. Transformer on the other hand completely forgets all the information that is outside of its context-window and it is not feasible to fit most datasets entirely in memory. Hence, we fix a certain roll-out length T defined in terms of the number of sequences not by the number of tokens. Also, we split the vocabulary $\mathcal{V}$ into the input $V_{\mathcal{X}}$ and output symbols $V_{\mathcal{Y}}$ ($V_{\mathcal{X}} \cup V_{\mathcal{Y}} = \mathcal{V}$, $V_{\mathcal{X}} \cap V_{\mathcal{Y}} = \emptyset$), and only define the permutations for the output symbols (i.e. $g(V_{\mathcal{X}}) = V_{\mathcal{X}}$)). Finally, for the sequence to sequence datasets, we do not need to predict the input sequence τ_x, hence we simply make the model predict a special token "PAD" for these portions.

Then, we can define the memory-based meta-learning algorithm for sequence to sequence datasets described in pseudo-code as Algorithm 2 (see Figure 3a for illustration). We first define the task distribution $\mathcal{G}_{\mathcal{V}}$ by generating R number of permutations of the vocabulary and O number of permutations of the samples $\mathcal{G}_D$ (i.e. rearrangement of dataset order). From here on, in order to avoid confusion, we refer to the permutation of labels as **permutation** and the permutation of samples as **reordering**. Then we construct our meta-episode by sampling N different permutations from $\mathcal{G}_{\mathcal{V}}$ and $\mathcal{G}_D$ independently. For each task, we reorder the dataset and rearrange the labels accordingly run our trajectory for T number of sequences. Hence, we use the term **trajectory** synonymously with a task. All the loss from N trajectories are accumulated and the accumulated loss is used to improve the meta-learner. This is repeated until the convergence of the learner.

[4]Note that this way can be considered defining a special permutation function g^* on an extended vocabulary $\hat{\mathcal{V}} = \mathcal{V} \cup \mathcal{W}$

Our method directly trains for generalization as each example can occur only once in each trajectory. Hence, the model needs to successfully bind the previously provided input-output pairs and use it for generalization. This is especially true when we permute the vocabulary to generate the task distribution as the model cannot memorize the dataset in its weights ($R > 1$). In other words, with $O = |D|!$, it trains on all possible k-shot learning problems that can be sampled from the dataset with the maximum k equal the maximum roll-out length T.

Algorithm 2 Memory-based Meta-Learning by Permutation of Samples and Labels

Given a sequence-to-sequence dataset $D = \{(\tau_x^{(i)}, \tau_y^{(i)})\}_{i=1}^{|D|}$, vocabulary $\mathcal{V} = V_{\mathcal{X}} \cup V_{\mathcal{Y}}$, batch-size N, maximum roll-out length T, number of permutations R, number of re-orderings O, meta-learner f
Generate R permutations of vocabulary $\mathcal{G}_{\mathcal{V}} = \{g^{(i)}\}_{i=1}^{R}$ where $g(\mathcal{V}_{\mathcal{X}}) = \mathcal{V}_{\mathcal{X}}$
Generate O permutations of samples $\mathcal{G}_{\mathcal{D}} = \{d^{(i)}\}_{i=1}^{S}$
while not converged **do**
    $L \leftarrow 0$
    for $n = 1$ to N **do**
        Sample a permutation $g^{(n)} \sim \mathcal{G}_{\mathcal{V}}$
        Sample an ordered dataset $\hat{D} = d(D), d \sim \mathcal{G}_{\mathcal{D}}$
        Initialize the starting memory state $m_0^{(n)}$
        for $t = 1$ to T **do**
            Sample t^{th} example from $\hat{D}$: $(\tau_x^{(t)}, \tau_y^{(t)})$
            Concatenate using special delimiters : $\tau^{(t)} = [IN, \tau_x^{(t)}, OUT, \tau_y^{(t)}]$
            Define a target sequence with shifted labels $\hat{\tau}^{(t)} = [PAD, ..., PAD, \tau_y^{(t)}, IN]$
            for $j = 1$ in $|\tau^{(t)}|$ **do**
                L $\leftarrow$ L + cross-entropy-loss($\hat{\tau}_j^{(t)}$, $f(\tau_{j-1}^{(t)}, m_{j-1}^{(t)})$)
                Update memory $m_j^{(t)}$
    minimizationStep(f, L)

4.4 Evaluation

Now we define how the network is evaluated on the test distribution. As seen before, compositional splits are designed for the zero-shot generalization. This means that the model is required to generalize to the test examples only by using the train examples. Hence, we simply use a random trajectory sampled from the dataset to condition the generalization on the test set. This is done by replacing the test example with the last sample of the sampled trajectory and evaluating. The evaluation of each test example is done independently using different randomly sampled trajectories (see Figure 3b for illustration). Note that if the trajectory length is equal to the dataset length, then this amounts to directly conditioning the test generalization on entire training dataset.

4.5 Theoretical Limit on Generalization Capability

Even though our method trains directly for generalization by discovering the structure behind the data distribution, this does not guarantee that the model will be able to discover the true data generative process of the data distribution. This is because this setup is naturally sensitive to frequency distribution of the train dataset in terms of the kinds of k-shot learning problems it contains. The number of possible k-shot learning problems is large, but it is still finite and incomprehensive and the model is able to overfit to these problems.

To better illustrate this limitation, suppose the underlying sequence distribution where the dataset was sampled from is a hidden markov-chain with a finite set of latent parameters and non-uniform state transition probabilities. Due to the non-uniform state transition probabilities, some latent trajectories are bound to be more frequent than others (refer to these as $\mathcal{Z}$). Suppose that the one resulting dataset from this distribution is dominated by the surface realizations of $\mathcal{Z}$. Then the model can be successful on this dataset by approximating the latent parameter $\mathcal{Z}$ as it is the latent parameters that are shared between most of the examples in the dataset. However, this would only captures one subset of of the transition probabilities and multiple approximations of $\mathcal{Z}$ are possible, the approximated representation of $\mathcal{Z}$ can overfit and might not generalize well to the other transitions under the same distribution.

5 Experiments and Results

In this chapter, we present empirical evidence that our memory-based meta-learning framework that simulates an online learning environment can lead to the compositional generalization capability to emerge in Transformer. We aim at understanding how our proposed approach can lead to compositional generalization by evaluating on difficult compositional splits of SCAN (Lake and Baroni, 2017). We achieve this by controlling the property of data distribution and performing ablative studies on our proposed approach.

In Section 5.1, we introduce the datasets utilized for evaluation, including the well-known SCAN dataset and its associated compositional splits. Additionally, we introduce two smaller datasets, AA and AATT, which we have designed to mimic the primitive generalization split of SCAN. Then in Section 5.2, we provide comprehensive details on the processing of the dataset, training configuration, and evaluation methodology employed in our study. Then we evaluate our approach through a series of four experiments from Section 5.3 to Section 5.6 each designed to elucidate different research questions regarding the emergence of compositional generalization. Each experiment section will provide an overview of its main motivation, research question, and specific experimental details followed by the results.

We concisely outline the key empirical findings that emerged from our research as follows:

- Our memory-based meta-learning framework leads a large improvement in compositional generalization in all compositional splits that we have evaluated on.
- Transformer is sensitive to the kind of generalization problems contained implicitly in the online trajectories.
- In-context learning can be learned even when in-weights learning is possible for Transformer.
- The emerged in-context learning ability is general to a degree, which allows Transformer to learn from trajectories inherently different than the ones seen during training.

5.1 Datasets

In this section, we provide details on three compositional generalization datasets used for our experiments: SCAN, AA, and AATT.

5.1.1 SCAN

Simplified version of the **C**omm**AI** **N**agviation Task (SCAN) is a text-to-text dataset where artificial language commands need to be mapped to a sequence of actions. The input commands are generated using a phrase structure grammar without recursion. Each command is mapped to exactly one correct sequence of actions using a set of semantic interpretation functions (see Table 6 for examples and Table 9 for the complete grammar and semantic functions). The words of the commands can be categorized into the *primitives* and *functions*. *Primitives* are the command words that have a direct counterpart in the action space such as "jump" which gets mapped to the action "JUMP". The primitives are modified through

the modifier functions such as "twice" and combined with conjunctions such as "after" into compounds compositionally. The dataset is generated by enumerating all possible strings that can be generated by the grammar which results in 20,910 sequence pairs.

Command	**Action Sequence**
jump	JUMP
run	RUN
run twice	RUN RUN
jump after run	RUN JUMP
jump around right	JUMP RTURN JUMP RTURN JUMP RTURN JUMP
jump around right after run twice	RUN RUN RTURN JUMP RTURN JUMP RTURN JUMP RTURN JUMP

Table 3: Example SCAN commands and their translated action sequences

This dataset can be systematically divided into the train and test set to test a particular type of compositional generalization. One such a split is the **add-jump-split** (SCAN-j) which is a primitive generalization task. In this task, the dataset is split in such a way that all the possible uses of the primitive "jump" is assigned to the test set except for its primitive mapping. Therefore, the model needs to generalize the use of functions to "jump" from the three other primitives. Hence, it tests for the systematicity of the model. Another way to split the dataset is through MCD principle (see Section 2.1.2) which attempts to make the compound frequency distribution of the train and test set divergent while maintaining the closeness of the atom frequency distribution. There exists three different MCD splits for SCAN (SCAN-MCD-n) indexed by the order of increasing divergence (i.e. SCAN-MCD-1 is the lowest and SCAN-MCD-3 is the highest) (Keysers et al., 2020). It was shown that the higher divergence score tightly correlates with the decrease in generalization performance in base deep learning architectures, which means that SCAN-MCD-3 is the most difficult MCD split. We refer the readers to Table 8 for the summary of the statistics of these splits and Appendix A.4 for more detailed explanation on how the MCD splits are actually generated in practice.

5.1.2 AA and AATT

The SCAN dataset is too large to fit into memory in its entirety. In order to test the setting where $T = |D|$, we create two smaller versions of SCAN: **A**and**A**fter (AA) and **A**nd**A**fter**T**wice**T**hrice (AATT). This is done by taking a subset of SCAN's context-free grammar rules and generating all possible commands without recursion. AA, as the name would suggest, takes the conjunctions "and" and "after" and the primitives to form the dataset. AATT adds an additional depth to AA with two more grammar rules using the modifiers "twice" and "thrice". Table 4 provides the entire grammar rules and the semantic interpretation functions for the two datasets.

For both dataset, we generate the primitive generalization compositional split by testing the model on alll possible uses of the primitive "jump" after seeing only the lone primitive mapping "jump $\rightarrow$ JUMP" during training. This results in a total of 124 tokens for AA and 1514 tokens for AATT which means that it becomes possible to fit the entire dataset in memory (i.e. $T = |D|$). The context length of 1514 tokens for AATT is not quite as large

as the context length of many LLMs but it is still big enough to study how the model can utilize the information with a large context.

Grammar	**Semantic Rules**
AndAfter	[jump] = JUMP
C → S	[run] = RUN
C → S and S	[walk] = WALK
C → S after S	[look] = LOOK
S → V	u and v = [u] [v]
V → U	u after v = [v] [u]
U → {jump, run, walk, look}	u twice = [u] [u]
AndAfterTwiceThrice	u thrice = [u] [u] [u]
S → V twice	
S → V thrice	

Table 4: AA and AATT dataset's phrase structure rules and semantic interpretation rules. AATT appends two additional grammar rules to that of AA. The bracket "[]" denotes the semantic interpretation converting the command into action sequences. u and v are variables and jump, look, run, and walk are primitives.

Dataset	**Train #Examples**	**Train #Tokens**	**Test # Examples**
AA	22	124	14
AATT	172	1514	28

Table 5: Dataset statistics of AA and AATT's primitive generalization splits.

5.2 Implementation Details

Here we provide general implementation details for our experiments to avoid redundancy. Unless noted otherwise, the setting detailed in this section will be used in the experiments.

5.2.1 Preprocessing

In order to decrease the sequence lengths of instructions of SCAN, we process the outputs (i.e. action sequences) using Python syntax as it was done in Wei et al. (2022b) for evaluating LLMs. This decreases the average output length of the dataset from 14.3 to 12.2 and the maximum sequence length from 48 to 17. In the new format, the overall vocabulary size of SCAN is 30 with 11 output words, 4 special symbols and 15 input words.

5.2.2 Model

We only use the encoder network of Transformer in our experiments. The encoder can be made causal by providing a triangular attention mask to prevent the position i from attending to the future positions of i. The variant where the LayerNorm is moved to the start of each sub-block is used as this is known to help stabilize the training of Transformers.

Command	Action Sequence
run twice	RUN * 2
jump after run	RUN + JUMP
jump around right	(RTURN JUMP) * 4
jump around right after run twice	(RTURN JUMP) * 4 + RUN * 2

Table 6: Example SCAN commands and their translated action sequences using Python syntax.

For all experiments, we use RAdam (Liu et al., 2020a) as the optimizer and use a linear warmup learning rate scheduler where the learning rate is linearly increased from 0 to the set learning rate for a specified number of steps at the start. We set the number of warm-up steps to 300 meta-episodes. These choices are important for the stabilization and convergence when training Transformers. Unless otherwise noted, we use the learning rate of 1×10^{-4} and momentum of (0.9, 0.99) for all of our experiments. Furthermore, we clip the norm of the gradient whenever it exceeds 5 and the use dropout rate of 0.1. We use absolute sinusoidal positional embedding (PE) of the original Transformer because we found that using learned PEs leads to less stable training. We train all of our models from scratch and do not experiment with using pretrained weights or word embeddings. We use the model dimension of 512 and feedforward dimension of 2048. We conduct all of our experiments using PyTorch (Paszke et al., 2019).

5.2.3 Task Distribution

Meta-learning is sensitive to the number of tasks and it usually requires a lot of tasks. General in-context learning algorithm is known to arise for a large number of tasks (Kirsch et al., 2022). How the task distribution is exactly defined will depend on each experiment, but we generally set the number of permutations R and number of reorderings O to the maximum possible number for the given vocabulary size and dataset respectively. This results in a sufficiently large task distribution as even with a small vocabulary size of 10 results in $10! \approx 2^{22}$ tasks. Finally, we sample $N = 32$ tasks (i.e. batch size) for AA and 6 for AATT. Lower batch size needs to be used for AATT to fit the model into memory.

5.2.4 Evaluation

The standard way of training a deep learning model for generalization on the test set of i.i.d samples is to keep a development set (i.e. dev set) for tuning the hyperparameters. This developement set is usually constructed by randomly splitting the training set into two parts. Then, the best checkpoint(s) of the model is chosen according to the developement set for evaluation on the test set. However, such a technique cannot be employed for compositional generalization as the higher performance on the training or dev set does not correlate well with higher performance on the test set (Csordás et al., 2021), because the test set is not distributed in the same way as the training or dev set. Often times, the test performance will continue to increase even when the training metric has converged. If we were to keep a portion of the test set for the purpose of using it as a dev set, then this would result in an

information leakage and tuning for a specific kind of generalization.

Our focus is not to find the state-of-the art method, but to evaluate how our MBML approach can lead to an emergent compositional generalization behaviour in deep learning models. Hence, instead of devising a new way of constructing a good dev set for OOD generalization, we simply train our model for a sufficient amount of steps and report on the converged test set scores. However, this is usually not reliable as we are still hand-selecting the hyperparameter for the number of meta-episodes as different configurations of the model lead to the model converging on the test distribution at different times. Hence, we mostly rely on presenting **test-score trajectories**, which graph the performance on (a subset of) the test set for different training checkpoints to be able to assess how the OOD generalization capability develops from the model.

When illustrating through a test-score trajectory, we use the mean of three runs with error bars using standard deviation. Finally, the plots are smoothed using exponential moving average for easier visualization as we find a large variation between different checkpoints. We also present the raw plots without any smoothing in Appendix B.

For all of our experiments, we present the sequence level accuracy (i.e. exact match) where the entire command needs to be correctly interpreted for it to be deemed correct. For zero-shot generalization on test examples, we generate the prediction by appending the test input as the T^{th} sample of the training trajectory of length $T-1$ and greedy-decoding auto-regressively . Finally, unless otherwise noted, all the presented models converge during training and is able to reach near 100% accuracy.

5.3 Experiment 1

In this experiment, we use the small datasets AA and AATT to test whether the compositionality would arise under the proposed meta-memory framework. Using these datasets allow us to study the condition where the rollout length T can be made the length of the entire dataset $|D|$ as both datasets are small enough to be able to fit into memory. Though they are small and simple, they are compositionally structured and the Transformer network is not able to discover a compositional solution by the standard training method.

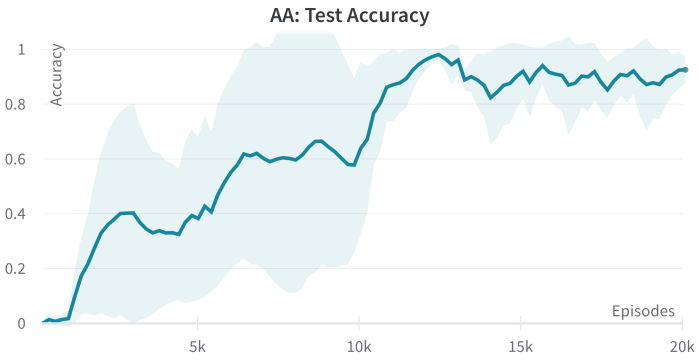
5.3.1 Experimental Setup

We construct our task distribution by allowing all permutations of the output labels and reorderings. For AA, there are 5 output types (i.e. the primitives and "+") which result in a total of 120 possible ways of assigning the input-output mappings. For AA, there are 7 outputs (i.e. adding "$*$", 2 and 3) which generate $5040 \approx 2^{14}$ tasks. For both datasets, we repeat the lone primitive example for "jump" four times as without it the model would not have any signal to memorize it[5]. There are 25! and 174! ways of rearrangements for AA and AATT. This means that the model will most likely never see the same task twice during training. Also, in this setting, in-weights learning is not possible as the input-output

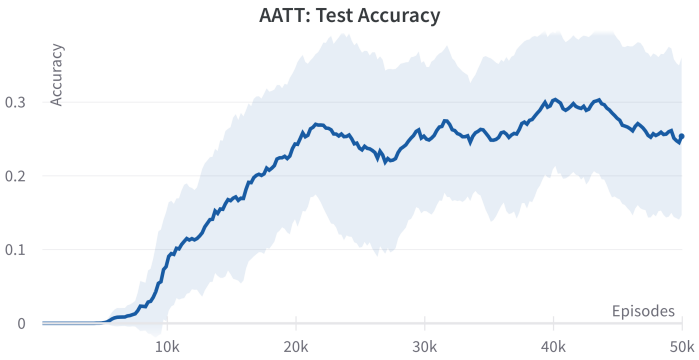
[5]The original SCAN-j configuration presented in Lake and Baroni (2017) also repeats the lone primitive mapping of "jump" to make the dataset more balanced in terms of primitive frequencies.

mappings are not fixed and each input can be mapped to any output labels. We train the model for 20k meta-episodes on AA and 50k on AATT using an 8-layer 4-head model. To create the test-score trajectory, we evaluate on the entire test set every 500 episodes.

5.3.2 Results



(a) Test Accuracy on AA



(b) Test Accuracy on AATT

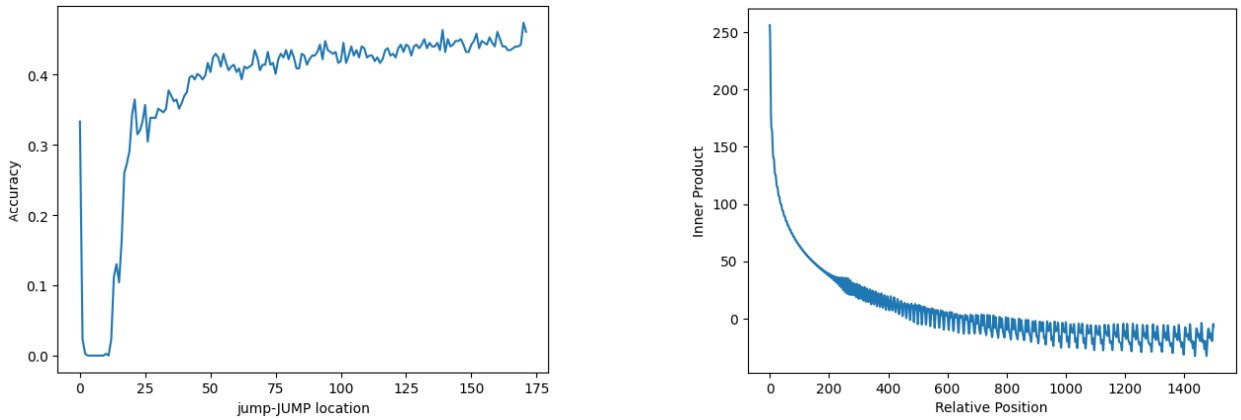
Figure 4: (a) plots the test-score trajectory on AA and (b) on AATT. For without smoothing, see Figure 14.

Figure 4 plots the results as test-score trajectories. Though not illustrated on the figure, on AA the model converges at around 3k and 10k for AATT. On AA, the model is able to compositionally generalize by successfully predicting over 95% of the test examples. On the other hand, the model's performance suffers greatly on AATT with only around 30% accuracy on the corresponding test set. Hence, we see that increasing the size of the context by ten-fold (from 124 to 1514) leads to a significantly lower generalization capability. This result is somewhat unexpected as Transformer is able to attend to the entire context in both cases using attention and does not suffer from forgetting like LSTM. Hence, the model *should* be able to generalize systematically in both cases by attending to the correct supporting examples in its past positions such as "jump $\rightarrow$ JUMP" for generalization.

We suspect that this drop in performance is due to the inductive bias of absolute sinusoidal PE which encodes for locality of reference. The inner product of the PEs decays as their

relative positional difference increases (see Figure 5b). This locality bias allows an easier modelling of natural language data where the neighboring tokens contain much more information than the distant ones[6]. Thus, this would result in a soft local context window where the model is biased towards preferring the use of support examples that are in closer proximity.

In order to test this hypothesis, we varied the location of "jump → JUMP" which the model needs for the successful generalization while shuffling the locations of the remaining examples. Figure 5a illustrates the result. We see a clear monotonically increasing trend in the model's performance as the primitive sample becomes closer to the test input. For some positions, the model is not at all able to generalize. We see that this plot closely resembles the inner products of PEs illustrated in Figure 5b. Finally, we see that even when the primitive mapping of "jump" is provided close to the test input, the model's performance maxes out near 50%, not being able to generalize successfully for half of the test set. Hence, these results suggest that the model does indeed have a (soft) local context window where the examples in close proximity are better utilized, but this can only account for one aspect for the model's sub-optimal compositionality.



(a) (b)

Figure 5: (a) The location of "jump → JUMP" is varied from the *sequence-level* position 0 (start) to 173 (end). Note that for evaluation, the test input is appended to the end of the training trajectory (at position 174). (b) Inner product of absolute sinusoidal positional embeddings for different relative *token-level* positions.

5.4 Experiment 2

The previous experiment illustrated that the absolute sinusoidal PE results in local context windows for Transformers and this causes the model to behave differently as the context window size increases. This would suggest that when training on long trajectories of sequences, the model has an inductive bias to perform *batch updates* on various spontaneously emerging

[6]This was empirically shown that the learned positional embeddings also show this locality of reference when trained on natural language data (Clark et al., 2019).

generalization problems within each trajectory. This enables us with a possible explanation for the remaining gap in the generalization performance on AATT.

We hypothesize that the low performance on AATT is due to the lack of the emergent primitive to compound generalization problems (prim→comp) it faces during training. In AATT, the primitives only consists 4% of the dataset, hence within each local context window, the model would rarely need to rely on using the primitive mapping such as "*look* → *LOOK*" to generalize for the compound such as "look thrice". Instead, the model would face significantly more problems using compounds to generalize to other compounds (comp→comp). In constrast to AATT, 30 % of AA consists of primitives which would ensure more prim → prim problems to arise during training. Lastly, notice that, in AATT, almost all examples that are in close proximity in the given trajectory are informative as the dataset was constructed by enumerating all possible strings under the grammar. Hence, due to this property of the data distribution, the model does not have any pressure to overcome the locality prior by being highly selective of which support examples to attend to.

Hence, we test these hypotheses through three separate experiments:

- **Experiment 2.1** Since the model faces more comp→comp problems during training, it should perform better when the lone primitive mapping of "jump" is replaced with the compound use of the primitive such as "jump and look → JUMP LOOK".
- **Experiment 2.2** We can ensure more prim→comp problems to emerge by changing the primitives to be more frequent in the dataset. Hence, the dataset with an increase number of the primitive mappings should ensure more local context windows to contain the primitives to use as support examples for generalization.
- **Experiment 2.3** The PE leads to the model being trained on different emergent generalization problems including the prim → comp problem. Then, replacing the PE with a different PE without any inductive bias for locality would lead to a decreased performance as the model can attend to all the past examples which are mostly compounds.

5.4.1 Experimental Setup

Generally, we use the same experimental setup as Experiment 1 by permuting all output labels and allowing all possible reorderings. We use an 8-layer 4-head model with $N = 6$.

Experiment 2.1. We construct an alternate training and test split for AATT where the lone primitive mapping "jump → JUMP" is swapped with "jump and look → JUMP LOOK". This alternate compound generalization split has only one example of "jump" in the training as before with the same number of training and test samples. We train for 100k episodes as we found that increasing the number of episodes can lead to an improvement in performance in this setting.

Experiment 2.2. We generate two more versions of AATT with an increased number of primitive mapping samples (i.e. jump, walk, look and run) so that each mapping can occur M times. We test for $M = 14$ and $M = 24$ which corresponds to the proportion of the

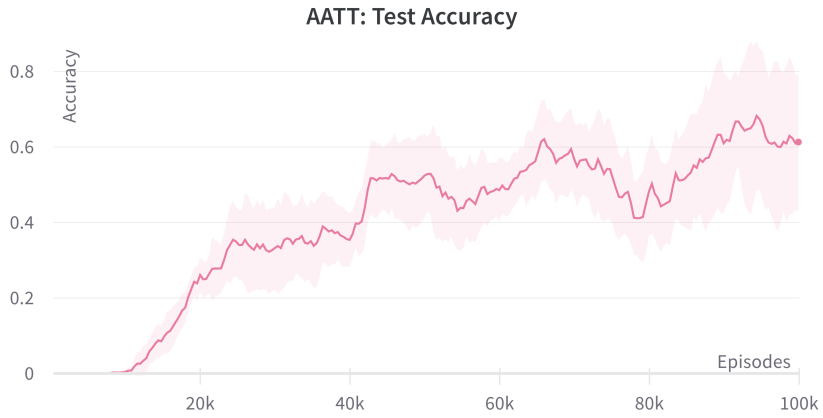


Figure 6: The plot illustrates the 8-layer 4-head Transformer model trained on the compound generalization split of AATT (without smoothing, see Figure 15).

primitive samples in the dataset to be increased to 10% and 34% respectively. Note that we are not adding new primitive types, but we are simply increasing the frequency of the existing primitive samples in the dataset. This would result in an increase of primitive mappings in each limited context window, hence increasing the number of prim→comp problems. We train on the new datasets for 50k episodes as in Experiment 1 (Section 5.3).

Experiment 2.3. We design a new PE that would allow the model to more easily attend to all the past examples in a trajectory. We do this by providing only the local positions within each sequence. For example, given the context [IN, jump, OUT, JUMP, IN, look, and, run, OUT, LOOK, +, RUN], the local positions for the PEs would correspond to [0, 1, 2, 3, 0, 1, 2, ..., 7]. Hence, this model would process the past examples as a set rather than a sequence. With this setup, the model would still be able to predict by generating for the lone-input.

5.4.2 Results

Experiment 2.1. The result of the model trained on the compound generalization split of AATT is illustrated in Figure 6. The model is able to generalize far better (∼100% relative improvement) than before by simply having a compound use of "jump". Since the model is trained on the comp→comp problems, it is able to generalize far better when it has "jump and look" in its context to generalize to other compounds such as "run after jump".

Experiment 2.2. Figure 7 shows the result of increasing the frequencies of primitives in AATT. As expected, we observe that increasing the number of primitive mapping samples leads to a drastic increase in generalization performance as it resulted in an increase of prim→prim problems.

Experiment 2.3. This resulted in the model losing all of its generalization capability (∼1%). The model was still able to reach near perfect accuracy (∼99%) on the training tasks. Upon qualitative evaluation on the generated outputs, the model produced the correct output

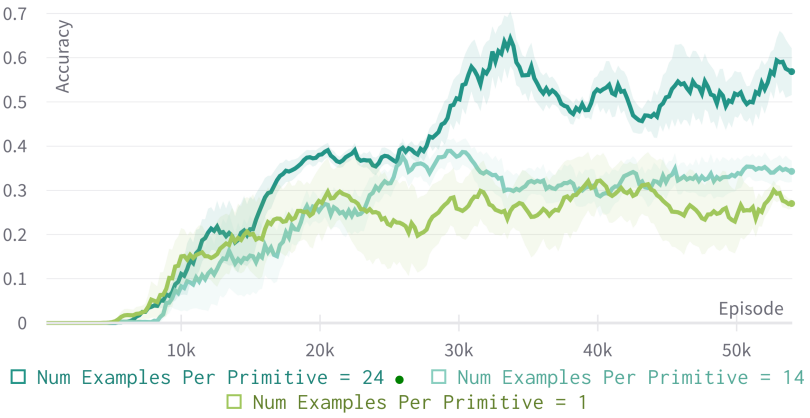


Figure 7: The primitive mappings were repeated M (i.e. Num Examples Per Primitive in the plot) times in the dataset for $M = 14$ and $M = 24$ (See Figure 16 without smoothing)

except for the primitive "jump". Since the model can attend to essentially all the previous examples and the compounds make up the overwhelming majority of the dataset, the model can be successful without needing to learn prim→comp during training.

These results confirms our hypothesis that model is sensitive to the generalization problems that spontaneously emerge from the online framework. Since Transformer is sensitive to the information available in its context, its generalization performance depends on the kinds of latent structure available in the provided set of support examples and how they need to be utilized for prediction. If it has not been trained on the type of generalization problem we require for the model to perform at test time, the model will not be able to generalize.

5.5 Experiment 3

From the previous two experiments, we have established how the different generalization problems can arise in our meta-memory learning framework that simulates an online learning environment. Especially, we found that the use of absolute sinusoidal PE was central for this emergent compositionality.

These insights allow us to observe that it is possible to interpret the addition of absolute sinusoial PEs to the word embeddings of a trajectory as a unique task. In other words, the prediction for the i^{th} sequence is not invariant to the *order* in which the previous examples arrived. This is because different orderings of the same set of sequences will be represented as a unique set of vectors due to the PEs. In the eyes of Transformer which processes all the inputs in parallel,. each trajectory will be a unique sequence, this would result in a distribution that is very large and diverse.

Then the training on a task distribution constructed by reorderings using absolute sinusoidal PE satisfies the conditions to be the hypothetical context-sensitive meta-learner discussed in Section 4.1. Such a meta-learner when learning on a highly-informative trajectories should be able to discover the in-context learning solution even when the in-weights learning is possible especially when the task distribution is not easy to memorize.

Hence, in this experiment, we investigate this hypothesis with the MCD splits of SCAN *without* the permutations of labels. These splits are bigger and more complex than the primitive generalization tasks of AA and AATT and mostly importantly do not require the existence of a particular support example for generalization (i.e. the primitive *jump* mapping). Hence, these splits can ensure the condition of highly informative trajectories for both training and evaluation. Also, since the MCD splits are bigger than AATT, the size of the resulting task distribution would be exponentially larger ensuring a more diverse distribution. Finally, the MCD splits tests for the more general kind of systematic generalization than primitive generalization as discussed in Section 2.1.4. Hence, these splits can truely test how much of the ground latent structure behind both train and test can be learned using our meta-memory framework from the training trajectories alone.

MCD splits are too big to be fit entirely into memory and we must specify the roll-out length T ($< |D|$) for the trajectories. $T = 1$ would equal to learning without meta-memory learning which we use this as the baseline. We should expect higher generalization performance for higher T because the effective number of reorderings is larger [7] and would lead to more emergent tasks.

5.5.1 Experimental Setup

We test on all three MCD splits available for SCAN. We vary the roll out length T to be $\{1, 2, 5, 10, 25, 50\}$ for MCD1 and $\{1, 25, 50\}$ for MCD2 and MCD3. We allow all possible reorderings of the dataset and do not permute any labels. This results in a very large number of tasks as the dataset consists of over 10k examples. We train an 8-layer 8-head model for 25k meta-episodes evaluating every 200 episodes on a randomly selected 10% of test set. We

[7] The effective number of reordering for a given T for the dataset of size $|D|$ is equal to $\frac{|D|!}{(|D|-T)!}$

also present the evaluation results for the main results from the selected checkpoints along with the test-score trajectory. Even though we do not evaluate on the entire test set, we found that evaluating on 10% of the test set is representative of the model's test performance as we sample different subsets each time and the test distribution is not long-tailed.

5.5.2 Results

Figure 8, 9, and 10 illustrate the results on different lengths of trajectories for MCD1, MCD2, and MCD3 respectively. Table 7 summarizes the evaluation result on MCD1 using the full test test for selected checkpoints. On all of these challenging compositional generalization splits, for all values of $T > 1$, the meta-learning Transformer is able to outperform the baseline. As expected, the test performance improves with higher T with $T = 50$ being the best model, improving the accuracy over the baseline by over 40% on MCD1. This trend is more clearly illustrated for MCD1 in Figure 8 where we see an increasing trend of average test accuracy scores upon convergence as the lengths of the trajectories are increased from 2 to 50.

Trajectory Length	**Accuracy**
1	22.5 ± 6.0
2	32.4 ± 15.8
5	27.9 ± 2.1
10	41.2 ± 10.6
25	37.8 ± 7.7
50	56.7 ± 14.3

Table 7: Models trained using different trajectory length are evaluated at 14k checkpoints on the full MCD1 test set. Trajectory length = 1 refers to the baseline without meta-memory learning.

Even though the MCD splits are constructed with varying difficulty with MCD1 being the easiest and MCD3 the hardest, we do not notice a noticeable drop in performance for our models between these splits. However, we do notice that the more difficult the split is, the faster the test performance falls towards the baseline performance as the model trains for longer. This is most likely due to the model over-fitting on the type of generalization problems that exists in their respective training sets. Since the splits are constructed by making the compound distributions of the train and test to diverge, higher divergence might indicate the higher risk of over-fitting as the model sees more and more tasks. This observation also provides us with a possible explanation on why the model is not able to perfectly generalize on the test set. As illustrated in the previous experiments, the model is sensitive to the kinds of k-shot learning problems that are implicit in different reorderings of the dataset. This means that the generalization problems frequent in the generated task distribution does not constitutes all the generalization problems that are not needed for the test set.

These results confirm our hypothesis that our MBML framework can lead to an improved generalization even when the in-weights learning is possible. However, the fact that the models are trained using MBML is not sufficient us to determine that the improved compositional

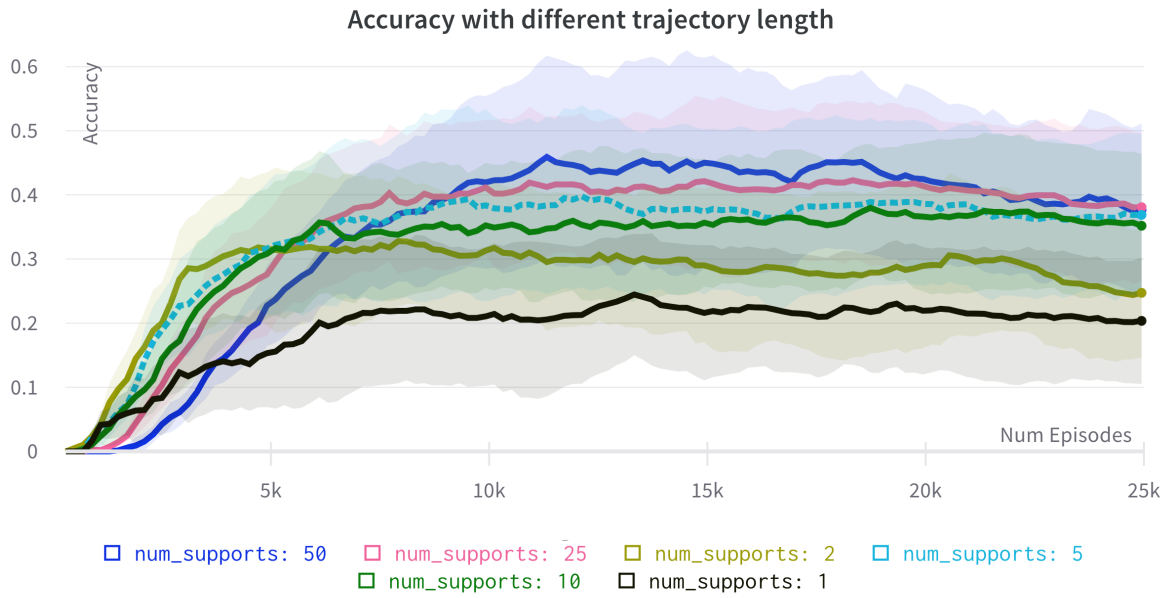


Figure 8: Test-score trajectory for MCD1 for different choices of trajectory lengths (i.e. num supports in the legend). See Figure 17 without smoothing.

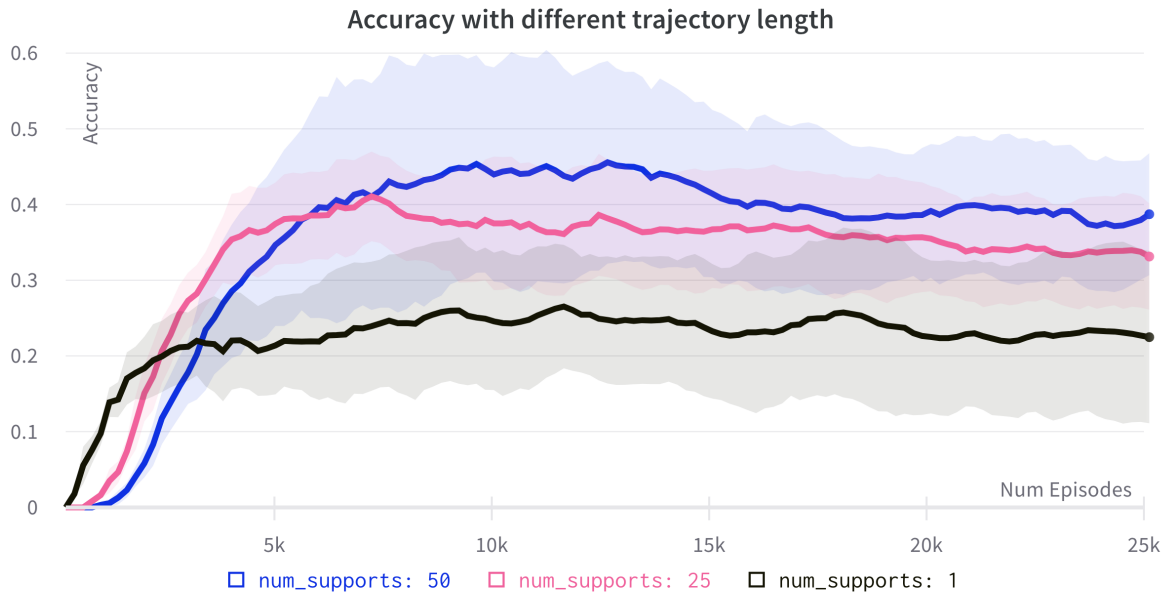


Figure 9: Test-score trajectory for MCD2 for different choices of trajectory lengths (i.e. num supports in the legend). See Figure 18 without smoothing.

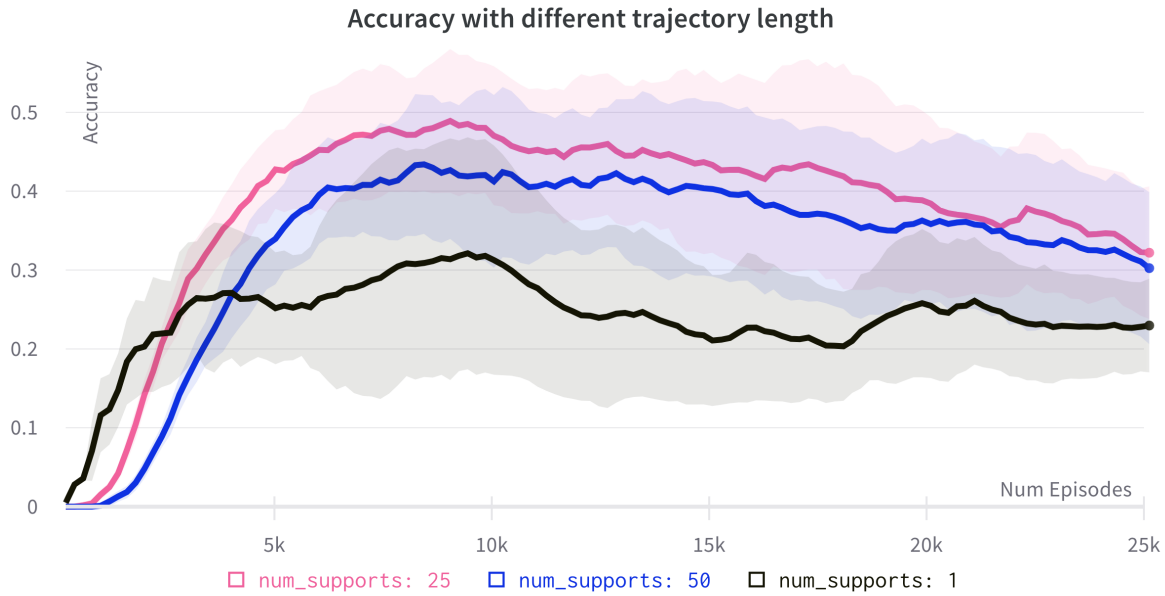


Figure 10: Test-score trajectory for MCD3 for different choices of trajectory lengths (i.e. num supports in the legend). See Figure 19 without smoothing.

generalization performance of the Transformer network is actually related to in-context learning.

We test this by using the $T = 50$ model and decreasing the number of support examples M by the increments of ten. More specifically, we test the model on different number of support samples: $M \in \{49, 40, 30, 20, 10, 1\}$. An in-context learning model would show a decrease in performance as less number of support examples are provided. Figure 11 presents the results. We see a clear increasing trend in the generalization performance as the number of support examples are increased. Thus, the in-context learning ability has emerged from the model even though in-weights learning was possible and this led to the model able to to better compositionally generalize.

In summary, under the online learning MBML framework, the Transformer network with absolute sinusoidal PEs is able to develop in-context learning for compositional generalization and this emergence occurs even when in-weights learning is possible. Its ability to learn from examples is used to model the latent structure underlying the examples provided in its context for OOD generalization.

5.6 Experiment 4

The results from Section 5.5 have demonstrated that under our meta-memory framework, in-context learning ability emerges and it allows improved generalization to the test set. We are interested in knowing how general this learned in-context learning ability is and we design two experiments to investigate this question:

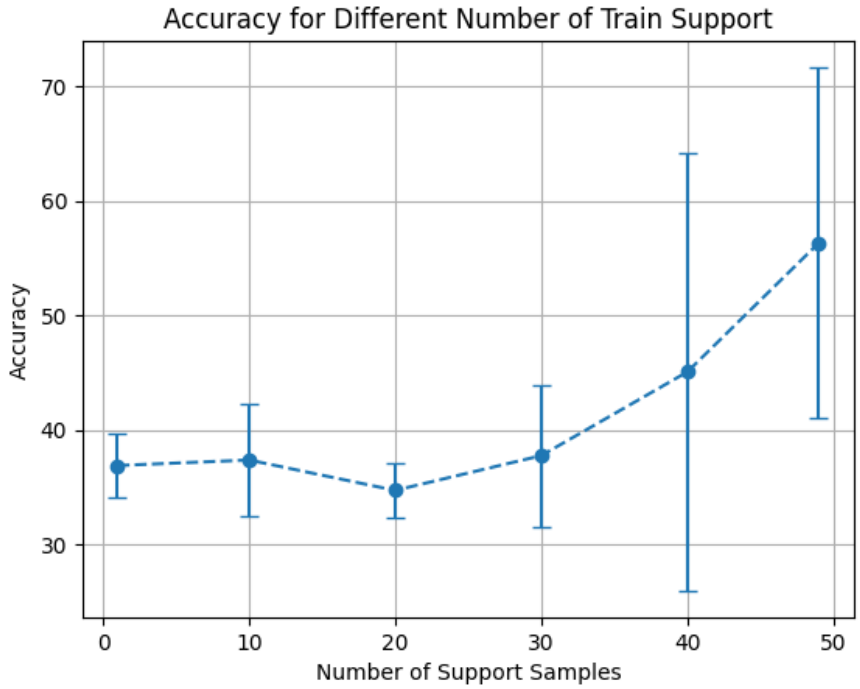


Figure 11: Number of support examples (i.e. M) was decreased by increments of 10 for the $T = 50$ model during evaluation. The plotted numbers for the support samples are $\{49, 40, 30, 20, 10, 1\}$.

- **Experiment 4.1.** We test the ability of the model's to utilize examples sampled from a different distribution than the one it was trained on. We can do this by sampling from the corresponding MCD test distribution as it is differently distributed from the training set by design. This can be done by splitting the test set randomly into two disjoint parts and using one split to generalize to another. Hence if the model has learned a general in-context learning capability, we should expect the model to improve in its performance as the test samples should be more informative.
- **Experiment 4.2** The in-context learning ability of LLMs are general as it is still able to generalize even when the support examples are provided using a special delimiter for separating the examples (discussed in Section 2.4.2). We test whether such an ability can also be found in our model by introducing sequences that would have been assigned zero-probability under the training distribution.

5.6.1 Experimental Setup

We use the $T = 50$ model trained on MCD1 for the experiments and evaluate on the entire test set.

Experiment 4.1. In order to split the test set into two disjoint parts, we randomly take M samples to be used as the support samples for evaluation and use the remaining examples as the new test set. We evaluate on different values of M to assess how the generalization

performance changes. In order to compare with the results of the model with different number of *train* support examples in Figure 11, we test on different number of *test* samples $M \in \{40, 30, 20, 10, 1\}$. Then we calculate the relative performance increase in percentage: $\frac{(new-before)}{before} * 100\%$.

Experiment 4.2. We generate the informative but unlikely trajectories by adding the sequence-delimiter "IN" additionally at the end of each sequence. The model understanding the role of a boundary marker should still be able to generalize properly. Hence, the new evaluation trajectories look as follows: $[IN, x^{(0)}, OUT, y^{(0)}, IN, IN, ..., x^{(M)}, OUT, y^{(M)}, IN, IN, x_q]$, where $(x^{(i)}, y^{(i)})$ are the input-output pairs from the train set and x_q is the test input. Note that we are using the standard way of sampling from the *train* set for zero-shot generalization. With this new support trajectories, we recreate the in-context learning experiment of Figure 11 by varying $M \in \{49, 40, 30, 20, 10, 1\}$.

5.6.2 Results

Experiment 4.1. Figure 12 plots the relative increase in performance when sampling from the test set. We see that for all values of M except for $M = 10$, the model is able to better generalize on their corresponding test set compared to using the train samples as the supports. However, the model is not able to drastically improve its performance even when a large number of test samples are provided (i.e. $M = 40$). This suggest that even though the model's in-context learning ability can be extended to learn from a different distribution, this is still limited by the generalization problems it has learned during training.

Experiment 4.2. Figure 13 plots the performance of the model for different number of support examples constructed as a trajectory with additional delimiters. Compared to the standard way of forming the trajectories, the generalization performance of the model suffers, but it is still able to better generalize than the baseline which scores 22.5% on the test set (see Table 7). We also observe the same increasing performance given more support examples which signifies the model is doing in-context learning. Hence, this result illustrates that the model is able to ignore the irrelevant delimiter token and learn from novel trajectories never seen during training. One possible reason for the decreased performance is that the model needs to generalize to a much longer sequence than it has experienced during training. This is because we add M additional tokens for the additional delimiters. Especially for higher M, this results in a large increase in the trajectory length.

These ablative studies have shown that the Transformer network from our framework is indeed able to do in-context learning and the ability to learn from experience is some what *general*. However, again, we observe that the model is limited by the k-shot learning problems implicit in the training distribution.

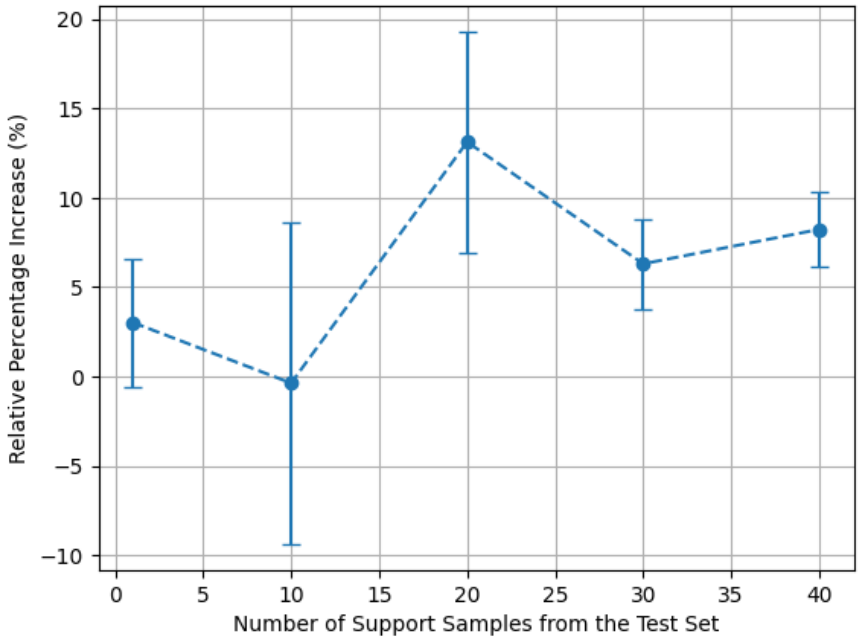


Figure 12: Number of support examples sampled from the MCD1 *test* set were decreased by increments of 10 for the $T = 50$ model during evaluation. Relative percentage increase is calculated with respect to sampling from the training set. The plotted numbers for the support samples are $\{40, 30, 20, 10, 1\}$.

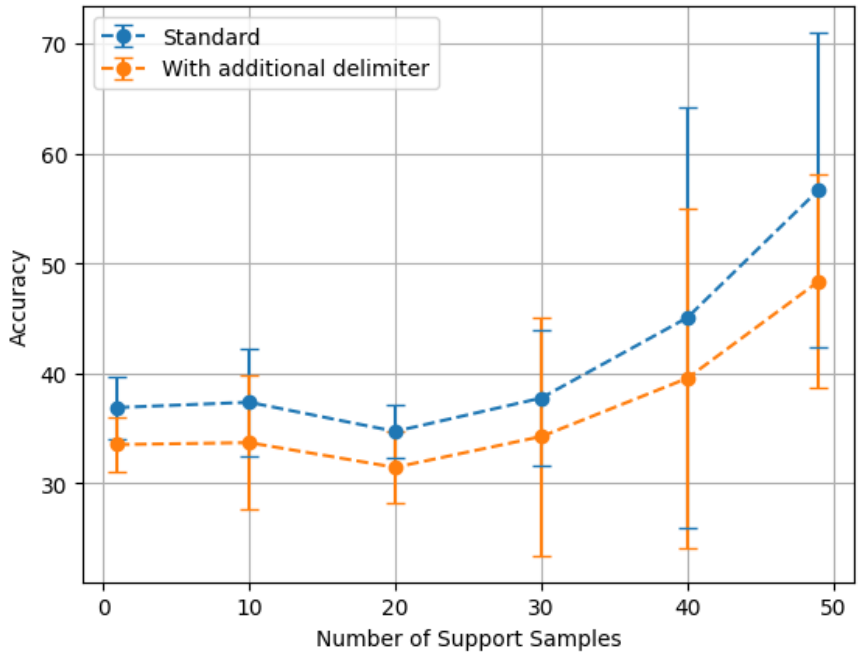


Figure 13: Plots $T = 50$ model evaluated on novel trajectories with additional delimiters (in orange). The standard approach is plotted in blue for comparison and it is replotted from Figre 11

6 Conclusion

The goal of the thesis was to study the relationship between in-context learning and compositional generalization. We accomplished this by training a model for in-context learning with memory-based meta-learning and evaluating on difficult sequence to sequence compositional generalization datasets.

We proposed a new meta-memory learning approach for training on sequence to sequence datasets for in-context learning and a novel evaluation framework for zero-shot compositional generalization. Our method was built on the intuition that different online learning trajectories of the same dataset can provide a wide range of generalization problems that can be transferred for compositional generalization. We devised two methodologies for constructing the task distribution of online trajectories for meta-learning, distinguishing between scenarios where in-weight learning is feasible and scenarios where it is not.

We evaluated our approach on three different compositional datasets including the widely known dataset called SCAN. The results showed that our method can lead to a significant improvement for compositional generalization. Especially, we have demonstrated that in-context learning and compositional generalization can arise even when in-weights learning is possible. Also, our additional experiments have showcased the sensitivity of the Transformer model to the few-shot learning problems that spontaneously arise in the online ordering of the dataset.

For future work, the possible directions include studying the relationship between the size of the model and the model's compositional generalization capability and evaluating our approach on other compositional datasets. Also, due to the computational constraints, we were only able to investigate up to 50 support examples for the MCD splits. Hence, it would be interesting to investigate how further increase of trajectory length can impact compositional generalization.

A Datasets for Testing Compositional Generalization

We provide more detailed information on the datasets presented in Section 2.3.1. Table 8 summarizes important statistics of each dataset.

Dataset	Train #	Test #	Tests for
SCAN-r	16.7k	4.1k	in-distribution
SCAN-j	14.6k	7.7k	primitive generalization (S)
SCAN-l	16.9k	3.9k	length generalization (P)
SCAN-mcd1	8.3k	1k	compound divergence (S + P)
SCAN-mcd2	8.3k	1k	compound divergence (S + P)
SCAN-mcd3	8.3k	1k	compound divergence (S + P)
PCFG-r	85k	10k	in-distribution
PCFG-s	82k	10k	function recombination (S)
PCFG-p	81k	11k	longer function combination (P)
COGS-r	24.1k	3k	in-distribution
COGS-g	24.1k	21k	linguistic genrealization (S + P)
CFQ-r	96l	12k	in-distribution
CFQ-mcd1	96k	12k	compound divergence (S + P)
CFQ-mcd2	96k	12k	compound divergence (S + P)
CFQ-mcd3	96k	12k	compound divergence (S + P)

Table 8: Summary statistics of the compositional generalization datasets are listed. "S" stands for systematicity and "P" for productivity.

A.1 SCAN

SCAN was properly introduced in Section 5.1.1, hence in this section, we only explain the compositional splits of the dataset included in Table 1. The random split (i.e. SCAN-r) is generated by the standard random 80/20 split of the entire dataset. *Length* split (i.e. SCAN-l) of SCAN tests productivity by testing the model on longer action sequences than the ones seen during training. The training set consists of all the commands with action sequences of length less than or equal to 22 and the test set consists of all the sequences with action sequences that are longer than 22 (i.e. 24 to 48). There exists three MCD splits for SCAN of varying difficulty with MCD1 being the easiest and MCD3 being the hardest.

A.2 PCFG

Probabilistic **C**ontext **F**ree **G**rammar SET (PCFG) dataset (Hupkes et al., 2020), as the name suggests, is generated using a set of phrase-structure grammar rules with recursion. It is a sequence to sequence dataset consisting of string operation tasks such as "repeat A B C $\rightarrow$ A B C". Similar to SCAN, the input to output interpretation function is fixed, hence a model that can discover this function can systematically generalize. There are two types of input words, *primitives* that are directly manipulated in producing the output string (i.e. A) and *functions* that defines the string edit operation (i.e. repeat). Unlike SCAN however,

Grammar	Semantic Rules
C → S	[jump] = JUMP
C → S and S	[run] = RUN
C → S after S	[walk] = WALK
S → V	[look] = LOOK
S → V twice	u and v = [u] [v]
S → V thrice	u after v = [v] [u]
V → D[1] opposite D[2]	u twice = [u] [u]
V → D[1] around D[2]	u thrice = [u] [u] [u]
V → D	[turn left] = LTURN
V → U	[turn right] = RTURN
D → U left	[u left] = LTURN [u]
D → U right	[u right] = RTURN [u]
D → turn left	[turn opposite right] = RTURN RTURN
D → turn right	[turn opposite left] = LTURN LTURN
U → {jump, run, walk, look}	[u opposite left] = [turn opposite left] [u]
	[u opposite right] = [turn opposite right] [u]
	[turn around left] = LTURN LTURN LTURN LTURN
	[turn around right] = RTURN RTURN RTURN RTURN
	[u around left] = LTURN [u] LTURN [u] LTURN [u] LTURN [u]
	[u around right] = RTURN [u] RTURN [u] RTURN [u] RTURN [u]

Table 9: Complete phrase structure grammarand semantic interpretation functions for SCAN. The bracket "[]" denotes the semantic interpretation converting the command into action sequences. u and v are variables and jump, look, run, and walk are primitives

PCFG generates the dataset probabilistically such that the depth and length distribution the dataset closely resembles the natural language distribution. The size of the alphabet is set to 520 and the dataset consists of 100k distinct input-output pairs.

The random in-distribution split (PCFG-r) of PCFG is generated by 85/5/15 split for the train, development, and test set respectively. The systematicity split of PCFG (PCFG-s) tests for the ability of the model to recombine functions and input strings never seen before during training. This is achieved by removing four pairs of functions to form the test set with the remainder of possible function pairs assigned to the train set. The productivity split of PCFG (PCFG-p) is constructed by withholding all sequences that contain more than eight functions from the training set.

A.3 COGS

A **CO**mpositional **G**eneralization Challenge Based on **S**emantic Interpretation (COGS) (Kim and Linzen, 2020) is a semantic parsing dataset that tests for different kinds of linguistic generalizations that humans are able to make. There are five different categories of linguistic generalization such as the primitive generalization of words seen as the subject to the object role or generalizing to deeper depths. The dataset is constructed using probabilistic context free grammar, but unlike the previous two datasets, the generated sentences resembles real natural language sentences. The dataset is able to capture some level of natural language

variation as the types of generated sentences can cover around 80% of naturally occurring English sentences. The sentences are mapped to their logical form using a simplified logical formalism. 30k distinct sentences are sampled consisting of 80 verbs and 60 nouns which are split using 80/10/10 split for the train, development, and test set. COGS only has one test split (COGS-g) which tests for all five categories of linguistic generalization and is generated by a separate probabilistic context-free grammar.

A.4 CFQ

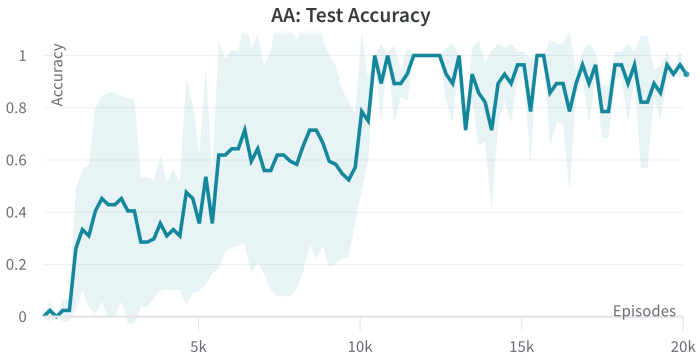
Compositional **F**reebase **Q**uestions (CFQ) (Keysers et al., 2020) is a semantic parsing dataset where the natural language instruction needs to be converted to SPARQL query. The dataset also provides natural language answer as well which allows this dataset to be used as a text-to-text task as well. The resulting SPARSQL query can be used to query from the public Freebase database (Bollacker et al., 2008) by replacing the output entities in terms of Freebase entity-ids. The dataset is generated automatically based on three kinds of rules: grammar, inference and resolution rules. Using the grammar rules, the natural language instructions and their corresponding logical forms are generated. Then, the inference rules and resolution rules are used to convert the logical forms into the queries. It consists of 239,357 distinct English input-output pairs.

CFQ was introduced along with the MCD principle which attempts to build a compositional split through increasing the divergence of the compound distributions while maintaining similar atom distributions between the train and test. Atoms are defined as individual rules in the directed acyclic graphs (DAG) describing the process of conversion from the natural language instruction to the SPARSQL query. Compounds are defined as sub-graphs of the DAGs. The MCD splits are generated using a greedy-algorithm that attempts to separate the dataset into two parts to achieve the desired divergence score. The random and MCD splits of CFQ are based on 40% and 10% of the whole dataset. There are three MCD splits of CFG with increasing divergence score (CFG-MCD1 to CFG-MCD2).

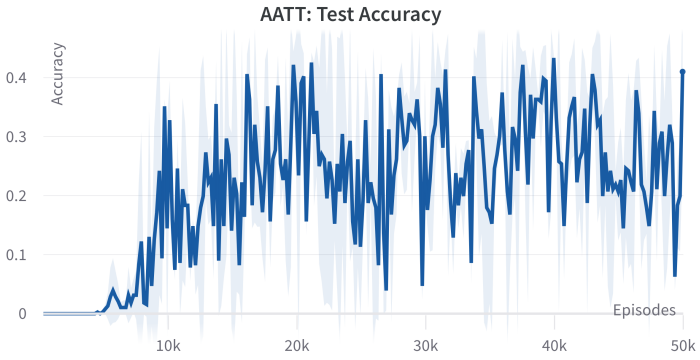
B Test-score Trajectories without Smoothing

In the experiment chapter, we presented the test-score trajectories presented from Section 5.3 to Section 5.6 smoothed using exponential moving average. In this section, we present the raw trajectories for the same figures without smoothing for completeness.

B.1 Experiment 1



(a) Test Accuracy on AA



(b) Test Accuracy on AATT

Figure 14: This plots shows the test trajectories of Figure 4 without smoothing. (a) plots the test trajectory on AA and (b) on AATT. .

B.2 Experiment 2

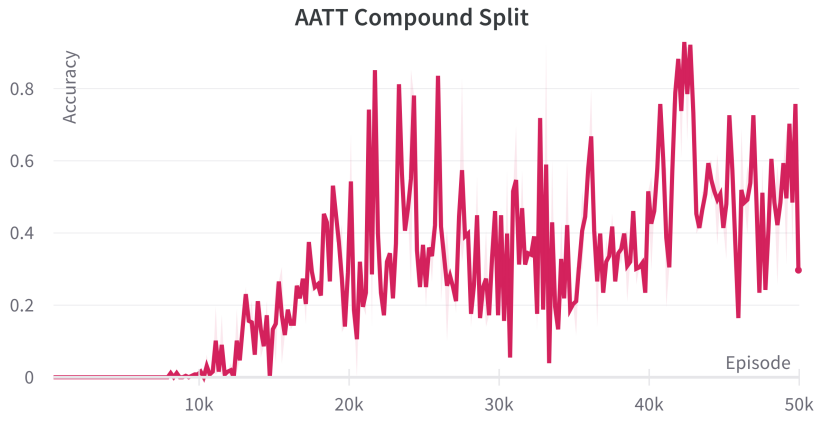


Figure 15: The plot illustrates the 8-layer 4-head Transformer model trained on the compound generalization split of AATT without smoothing.

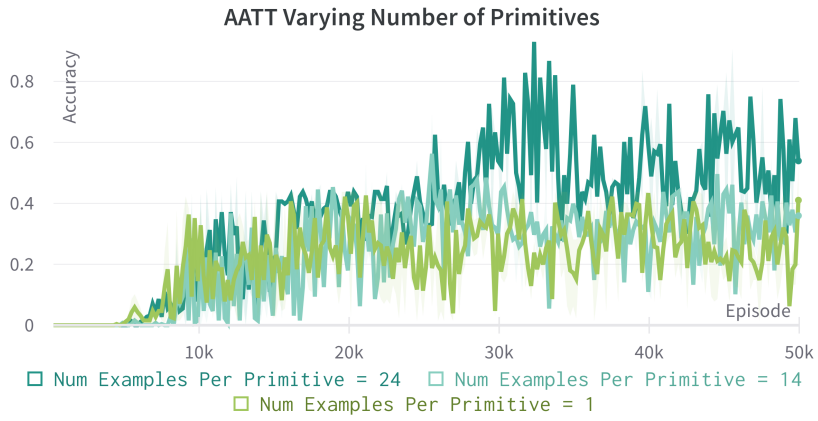


Figure 16: The primitive mappings were repeated M (i.e. Num Examples Per Primitive in the plot) times in the dataset for $M = 14$ and $M = 24$ (See Figure 7 with smoothing)

B.3 Experiment 3

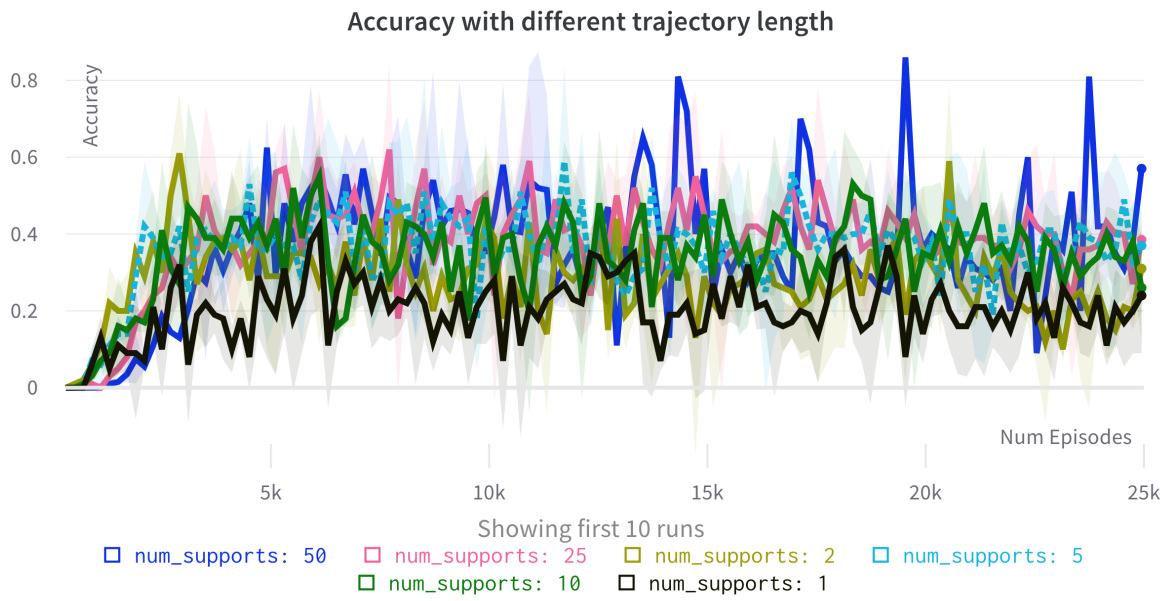


Figure 17: Test-score trajectory for MCD1 for different choice of trajectory length. See Figure 8 with smoothing.

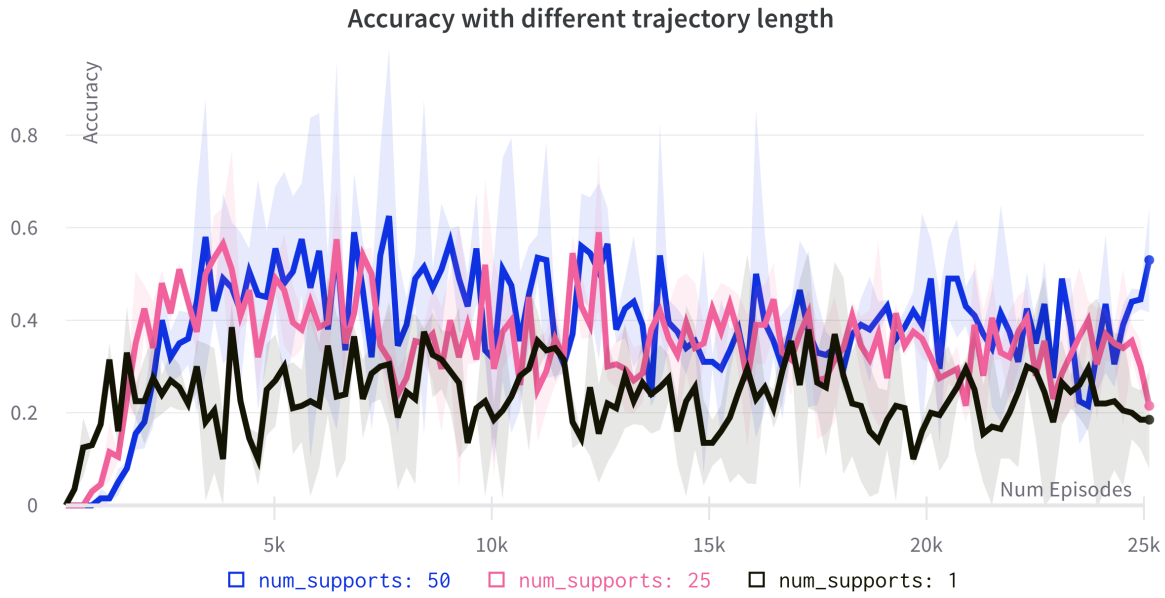


Figure 18: Test-score trajectory for MCD2 for different choice of trajectory length. See Figure 9 with smoothing.

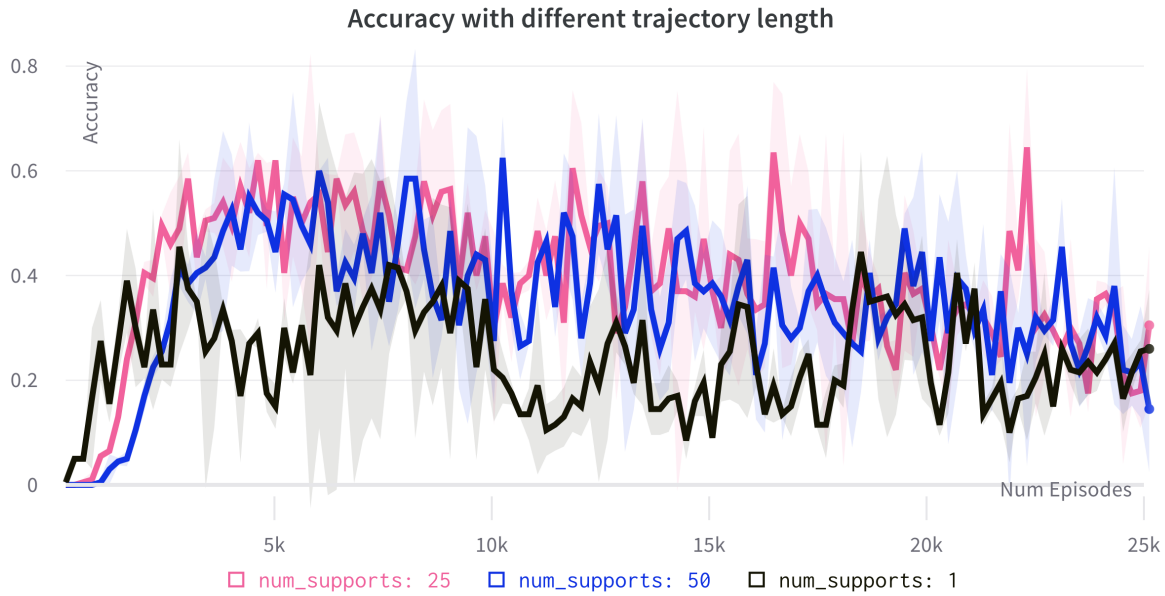


Figure 19: Test-score trajectory for MCD3 for different choice of trajectory length. See Figure 10 with smoothing.

List of Figures

1 Illustration of compositional split 8
2 Illustration of memory-based meta-learning 14
3 Illustration of the proposed memory-based meta-learning framework for compositional generalization 28
4 Test trajectories on AA and AATT 39
5 Varying location of "jump" and illustration of the inner-products of absolute sinusoial PE 40
6 Compound Generalization Split of AATT 42
7 Increasing the number of primitive samples in AATT 43
8 MCD1 with different roll out lengths 46
9 MCD2 with different roll out lengths 46
10 MCD3 with different roll out lengths 47
11 Testing Transformer model's in-context learning on MCD1 48
12 In-context learning with sequences from a different distribution 50
13 In-context learning with zero probability trajectories under the training distribution 50
14 Test trajectories on AA and AATT without smoothing 58
15 Compound Generalization Split of AATT without smoothing 59
16 Increasing the number of primitive samples in AATT without smoothing . . 59
17 MCD1 with different roll out lengths without smoothing 60
18 MCD2 with different roll out lengths without smoothing 60
19 MCD3 with different roll out lengths without smoothing 61

List of Tables

1 Lack of compositional generalization: Synthetic Text to Text Tasks 12
2 Lack of compositional generalization: Semantic Parsing Tasks 12
3 SCAN examples 35
4 Grammar rules and intepretation fucnctions of AA and AATT 36
5 AA and AATT primitive generalization split statistics 36
6 SCAN examples using Python syntax 37
7 MCD1 results on the entire test set 45
8 Compositional Split Statistics 54
9 Complete grammar rules and interpretation functions of SCAN 55

List of Algorithms

1 Memory-based Meta Learning for Sequential Predition Task 16
2 Memory-based Meta-Learning by Permutation of Samples and Labels 31

References

1. Agrawal, A., Kembhavi, A., Batra, D., and Parikh, D. (2017). C-VQA: A Compositional Split of the Visual Question Answering (VQA) v1.0 Dataset. arXiv:1704.08243 [cs].

2. Akyürek, E., Akyürek, A. F., and Andreas, J. (2021). Learning to recombine and resample data for compositional generalization. In *International Conference on Learning Representations*.

3. Akyurek, E. and Andreas, J. (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.

4. Altmann, E. G., Pierrehumbert, J. B., and Motter, A. E. (2009). Beyond Word Frequency: Bursts, Lulls, and Scaling in the Temporal Distributions of Words. *PLoS ONE*, 4(11):e7678.

5. Anderson, P. W. (1972). More is different. *Science*, 177(4047):393–396.

6. Andreas, J. (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.

7. Andrychowicz, M., Denil, M., Gómez, S., Hoffman, M. W., Pfau, D., Schaul, T., Shillingford, B., and de Freitas, N. (2016). Learning to learn by gradient descent by gradient descent. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

8. Anil, C., Wu, Y., Andreassen, A. J., Lewkowycz, A., Misra, V., Ramasesh, V. V., Slone, A., Gur-Ari, G., Dyer, E., and Neyshabur, B. (2022). Exploring length generalization in large language models. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K., editors, *Advances in Neural Information Processing Systems*.

9. Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

10. Bahdanau, D., Murty, S., Noukhovitch, M., Nguyen, T. H., de Vries, H., and Courville, A. (2019). Systematic generalization: What is required and can it be learned? In *International Conference on Learning Representations*.

11. Bengio, Y., Bengio, S., and Cloutier, J. (1991). Learning a synaptic learning rule. In *IJCNN-91-Seattle International Joint Conference on Neural Networks*, volume ii, pages 969 vol.2–.

12. Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 41–48, Montreal Quebec Canada. ACM.

13. Bergen, L., O' Donnell, T., and Bahdanau, D. (2021). Systematic generalization with edge transformers. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*, volume 34, pages 1390–1402. Curran Associates, Inc.

14. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., and Taylor, J. (2008). Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250, New York, NY, USA. ACM.

15. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

16. Calvo, P. and Symons, J. (2014). The architecture of cognition: Rethinking fodor and pylyshyn's systematicity challenge. *Philosophical Psychology*, 29(3):476–478.

17. Cann, R. (1993). *Formal Semantics: An Introduction*. Cambridge Textbooks in Linguistics. Cambridge University Press.

18. Chan, S. C. Y., Santoro, A., Lampinen, A. K., Wang, J. X., Singh, A., Richemond, P. H., McClelland, J., and Hill, F. (2022). Data Distributional Properties Drive Emergent In-Context Learning in Transformers. arXiv:2205.05055 [cs].

19. Chen, X., Liang, C., Yu, A. W., Song, D., and Zhou, D. (2020). Compositional generalization via neural-symbolic stack machines. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1690–1701. Curran Associates, Inc.

20. Chomsky, N. (1965). *Aspects of the Theory of Syntax*. The MIT Press, Cambridge.

21. Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A., Barnes, P., Tay, Y., Shazeer, N., Prabhakaran, V., Reif, E., Du, N., Hutchinson, B., Pope, R., Bradbury, J., Austin, J., Isard, M., Gur-Ari, G., Yin, P., Duke, T., Levskaya, A., Ghemawat, S., Dev, S., Michalewski, H., Garcia, X., Misra, V., Robinson, K., Fedus, L., Zhou, D., Ippolito, D., Luan, D., Lim, H., Zoph, B., Spiridonov, A., Sepassi, R., Dohan, D., Agrawal, S., Omernick, M., Dai, A. M., Pillai, T. S., Pellat, M., Lewkowycz, A., Moreira, E., Child, R., Polozov, O.,

Lee, K., Zhou, Z., Wang, X., Saeta, B., Diaz, M., Firat, O., Catasta, M., Wei, J., Meier-Hellstern, K., Eck, D., Dean, J., Petrov, S., and Fiedel, N. (2022). PaLM: Scaling Language Modeling with Pathways. arXiv:2204.02311 [cs].

22. Chowdhury, S. A. and Zamparelli, R. (2018). RNN simulations of grammaticality judgments on long-distance dependencies. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 133–144, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

23. Clark, K., Khandelwal, U., Levy, O., and Manning, C. D. (2019). What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

24. Conklin, H., Wang, B., Smith, K., and Titov, I. (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.

25. Csordás, R., Irie, K., and Schmidhuber, J. (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.

26. Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q., and Salakhutdinov, R. (2019). Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy. Association for Computational Linguistics.

27. Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J., and Kaiser, L. (2019). Universal transformers. In *International Conference on Learning Representations*.

28. Dessì, R. and Baroni, M. (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.

29. Duan, Y., Schulman, J., Chen, X., Bartlett, P. L., Sutskever, I., and Abbeel, P. (2017). RLˆ2: Fast reinforcement learning via slow reinforcement learning.

30. Edunov, S., Ott, M., Auli, M., and Grangier, D. (2018). Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500, Brussels, Belgium. Association for Computational Linguistics.

31. Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14:179–211.

32. Finegan-Dollak, C., Kummerfeld, J. K., Zhang, L., Ramanathan, K., Sadasivam, S., Zhang, R., and Radev, D. (2018). Improving text-to-SQL evaluation methodology. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.

33. Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR.

34. Fodor, J. A. and Pylyshyn, Z. W. (1988). Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1):3–71.

35. Fortunato, M., Tan, M., Faulkner, R., Hansen, S. S., Badia, A. P., Buttimore, G., Deck, C., Leibo, J. Z., and Blundell, C. (2019). Generalization of reinforcement learners with working and episodic memory. In *Neural Information Processing Systems*.

36. Furrer, D., van Zee, M., Scales, N., and Schärli, N. (2021). Compositional Generalization in Semantic Parsing: Pre-training vs. Specialized Architectures. arXiv:2007.08970 [cs].

37. Garg, S., Tsipras, D., Liang, P. S., and Valiant, G. (2022). What can transformers learn in-context? a case study of simple function classes. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A., editors, *Advances in Neural Information Processing Systems*, volume 35, pages 30583–30598. Curran Associates, Inc.

38. Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. N. (2017). Convolutional sequence to sequence learning. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1243–1252. PMLR.

39. Goldberg, A. E. (2019). *Explain me this: creativity, competition, and the partial productivity of constructions*. Princeton University Press, Princeton, New Jersey. OCLC: on1032723071.

40. Gordon, J., Lopez-Paz, D., Baroni, M., and Bouchacourt, D. (2020). Permutation equivariant models for compositional generalization in language. In *International Conference on Learning Representations*.

41. Gu, J., Wang, Y., Chen, Y., Li, V. O. K., and Cho, K. (2018). Meta-learning for low-resource neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3622–3631, Brussels, Belgium. Association for Computational Linguistics.

42. Gulordava, K., Bojanowski, P., Grave, E., Linzen, T., and Baroni, M. (2018). Colorless green recurrent networks dream hierarchically. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics:*

Human Language Technologies, Volume 1 (Long Papers), pages 1195–1205, New Orleans, Louisiana. Association for Computational Linguistics.

43. Guo, Y., Lin, Z., Lou, J.-G., and Zhang, D. (2020a). Hierarchical poset decoding for compositional generalization in language. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6913–6924. Curran Associates, Inc.

44. Guo, Y., Zhu, H., Lin, Z., Chen, B., Lou, J.-G., and Zhang, D. (2020b). Revisiting iterative back-translation from the perspective of compositional generalization. In *AAAI Conference on Artificial Intelligence*.

45. Gómez, R. L. (2002). Variability and Detection of Invariant Structure. *Psychological Science*, 13(5):431–436.

46. Herzig, J., Shaw, P., Chang, M.-W., Guu, K., Pasupat, P., and Zhang, Y. (2021). Unlocking Compositional Generalization in Pre-trained Models Using Intermediate Representations. arXiv:2104.07478 [cs].

47. Hill, F., Lampinen, A., Schneider, R., Clark, S., Botvinick, M., McClelland, J. L., and Santoro, A. (2020). Environmental drivers of systematicity and generalization in a situated agent. In *International Conference on Learning Representations*.

48. Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

49. Hochreiter, S., Younger, A. S., and Conwell, P. R. (2001). Learning to learn using gradient descent. In *Proceedings of the International Conference on Artificial Neural Networks*, ICANN '01, page 87–94, Berlin, Heidelberg. Springer-Verlag.

50. Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366.

51. Hsu, J.-Y., Chen, Y.-J., and yi Lee, H. (2019). Meta learning for end-to-end low-resource speech recognition. *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7844–7848.

52. Hupkes, D., Dankers, V., Mul, M., and Bruni, E. (2020). Compositionality decomposed: How do neural networks generalise? In Bessiere, C., editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 5065–5069. International Joint Conferences on Artificial Intelligence Organization. Journal track.

53. Hupkes, D., Singh, A., Korrel, K., Kruszewski, G., and Bruni, E. (2019). Learning compositionally through attentive guidance. arXiv:1805.09657 [cs].

54. Hupkes, D. and Zuidema, W. (2018). Visualisation and 'diagnostic classifiers' reveal how recurrent and recursive neural networks process hierarchical structure (extended abstract). In *Proceedings of the Twenty-Seventh International Joint Conference on*

Artificial Intelligence, IJCAI-18, pages 5617–5621. International Joint Conferences on Artificial Intelligence Organization.

55. Jiang, Y. and Bansal, M. (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.

56. Johns, B. T., Dye, M., and Jones, M. N. (2016). The influence of contextual diversity on word learning. *Psychon Bull Rev*, 23(4):1214–1220.

57. Johnson, J., Hariharan, B., van der Maaten, L., Fei-Fei, L., Zitnick, C. L., and Girshick, R. B. (2016). Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1988–1997.

58. Keysers, D., Schärli, N., Scales, N., Buisman, H., Furrer, D., Kashubin, S., Momchev, N., Sinopalnikov, D., Stafiniak, L., Tihon, T., Tsarkov, D., Wang, X., van Zee, M., and Bousquet, O. (2020). MEASURING COMPOSITIONAL GENERALIZATION: A COMPREHENSIVE METHOD ON REALISTIC DATA. page 38.

59. Kim, N. and Linzen, T. (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.

60. Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

61. Kirsch, L., Harrison, J., Sohl-Dickstein, J., and Metz, L. (2022). General-Purpose In-Context Learning by Meta-Learning Transformers. arXiv:2212.04458 [cs, stat].

62. Kirsch, L., van Steenkiste, S., and Schmidhuber, J. (2020). Improving generalization in meta reinforcement learning using learned objectives. In *International Conference on Learning Representations*.

63. Klejch, O., Fainberg, J., Bell, P., and Renals, S. (2019). Speaker adaptive training using model agnostic meta-learning. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 881–888.

64. Koch, G., Zemel, R., and Salakhutdinov, R. (2015). Siamese neural networks for one-shot image recognition.

65. Korrel, K., Hupkes, D., Dankers, V., and Bruni, E. (2019). Transcoding compositionally: Using attention to find more generalizable solutions. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 1–11, Florence, Italy. Association for Computational Linguistics.

66. Lake, B. M. (2019). Compositional generalization through meta sequence-to-sequence learning. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

67. Lake, B. M. and Baroni, M. (2017). Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International Conference on Machine Learning*.

68. Lake, B. M., Ullman, T. D., Tenenbaum, J. B., and Gershman, S. J. (2018). Building machines that learn and think like people. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '18, page 5, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

69. Lakretz, Y., Kruszewski, G., Desbordes, T., Hupkes, D., Dehaene, S., and Baroni, M. (2019). The emergence of number and syntax units in lstm language models.

70. Lampinen, A., Dasgupta, I., Chan, S., Mathewson, K., Tessler, M., Creswell, A., McClelland, J., Wang, J., and Hill, F. (2022). Can language models learn from explanations in context? In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 537–563, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

71. LeCun, Y., Boser, B., Denker, J. S., Howard, R. E., Habbard, W., Jackel, L. D., and Henderson, D. (1990). *Handwritten Digit Recognition with a Back-Propagation Network*, page 396–404. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

72. Li, K. and Malik, J. (2017). Learning to optimize. In *International Conference on Learning Representations*.

73. Li, Y., Zhao, L., Wang, J., and Hestness, J. (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.

74. Li, Z., Zhou, F., Chen, F., and Li, H. (2017). Meta-sgd: Learning to learn quickly for few shot learning. *CoRR*, abs/1707.09835.

75. Linzen, T., Dupoux, E., and Goldberg, Y. (2016). Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

76. Liu, C., An, S., Lin, Z., Liu, Q., Chen, B., Lou, J.-G., Wen, L., Zheng, N., and Zhang, D. (2021). Learning Algebraic Recombination for Compositional Generalization. arXiv:2107.06516 [cs].

77. Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J., and Han, J. (2020a). On the variance of the adaptive learning rate and beyond. In *International Conference on Learning Representations*.

78. Liu, L., Liu, X., Gao, J., Chen, W., and Han, J. (2020b). Understanding the difficulty of training transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5747–5763, Online. Association for Computational Linguistics.

79. Liu, Q., An, S., Lou, J.-G., Chen, B., Lin, Z., Gao, Y., Zhou, B., Zheng, N., and Zhang, D. (2020c). Compositional Generalization by Learning Analytical Expressions. arXiv:2006.10627 [cs].

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

81. Loula, J., Baroni, M., and Lake, B. (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.

82. Lu, Y. and Lu, J. (2020). A universal approximation theorem of deep neural networks for expressing probability distributions. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 3094–3105. Curran Associates, Inc.

83. Marcus, G. F. (2001). *The Algebraic Mind: Integrating Connectionism and Cognitive Science*. The MIT Press.

84. McClelland, J. L., Botvinick, M. M., Noelle, D. C., Plaut, D. C., Rogers, T. T., Seidenberg, M. S., and Smith, L. B. (2010). Letting structure emerge: connectionist and dynamical systems approaches to cognition. *Trends in Cognitive Sciences*, 14(8):348–356.

85. Melo, L. C. (2021). Transformers are Meta-Reinforcement Learners.

86. Mishra, N., Rohaninejad, M., Chen, X., and Abbeel, P. (2018). A simple neural attentive meta-learner. In *International Conference on Learning Representations*.

87. Müller, R., Kornblith, S., and Hinton, G. E. (2019). When does label smoothing help? In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

88. Nichol, A., Achiam, J., and Schulman, J. (2018). On first-order meta-learning algorithms.

89. Nooralahzadeh, F., Bekoulis, G., Bjerva, J., and Augenstein, I. (2020). Zero-shot cross-lingual transfer with meta learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4547–4562, Online. Association for Computational Linguistics.

90. Norman, R., Hulme, R. C., Sarantopoulos, C., Chandran, V., Shen, H., Rodd, J. M., Joseph, H., and Taylor, J. S. H. (2022). Contextual diversity during word learning through reading benefits generalisation of learned meanings to new contexts. *Q J Exp Psychol (Hove)*, page 17470218221126976.

91. Nye, M. I., Solar-Lezama, A., Tenenbaum, J. B., and Lake, B. M. (2020). Learning Compositional Rules via Neural Program Synthesis. arXiv:2003.05562 [cs].

92. Ontañón, S., Ainslie, J., Cvicek, V., and Fisher, Z. (2022). Making Transformers Solve Compositional Tasks. arXiv:2108.04378 [cs].

93. Oren, I., Herzig, J., and Berant, J. (2021). Finding needles in a haystack: Sampling Structurally-diverse Training Sets from Synthetic Data for Compositional Generalization. arXiv:2109.02575 [cs].

94. Ortega, P. A., Wang, J. X., Rowland, M., Genewein, T., Kurth-Nelson, Z., Pascanu, R., Heess, N., Veness, J., Pritzel, A., Sprechmann, P., Jayakumar, S. M., McGrath, T., Miller, K., Azar, M., Osband, I., Rabinowitz, N., György, A., Chiappa, S., Osindero, S., Teh, Y. W., van Hasselt, H., de Freitas, N., Botvinick, M., and Legg, S. (2019). Meta-learning of Sequential Strategies. arXiv:1905.03030 [cs, stat].

95. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

96. Patel, A., Bhattamishra, S., Blunsom, P., and Goyal, N. (2022). Revisiting the Compositional Generalization Abilities of Neural Sequence Models. arXiv:2203.07402 [cs].

97. Popel, M. and Bojar, O. (2018). Training tips for the transformer model. *The Prague Bulletin of Mathematical Linguistics*, 110(1):43–70.

98. Prabhu Prakash, K. (2020). SYSTEMATIC GENERALIZATION EMERGES IN SEQ2SEQ MODELS WITH VARIABILITY IN DATA. *ICLR "Bridging AI and Cognitive Science"*.

99. Qian, K. and Yu, Z. (2019). Domain adaptive dialog generation via meta learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2639–2649, Florence, Italy. Association for Computational Linguistics.

100. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners.

101. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

102. Ravi, S. and Larochelle, H. (2016). Optimization as a model for few-shot learning. In *International Conference on Learning Representations*.

103. Ritter, S., Faulkner, R., Sartran, L., Santoro, A., Botvinick, M., and Raposo, D. (2021). Rapid task-solving in novel environments. In *International Conference on Learning Representations*.

104. Ritter, S., Wang, J., Kurth-Nelson, Z., Jayakumar, S., Blundell, C., Pascanu, R., and Botvinick, M. (2018). Been there, done that: Meta-learning with episodic recall. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4354–4363. PMLR.

105. Ruis, L., Andreas, J., Baroni, M., Bouchacourt, D., and Lake, B. M. (2020). A benchmark for systematic generalization in grounded language understanding. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 19861–19872. Curran Associates, Inc.

106. Russin, J., Jo, J., O'Reilly, R. C., and Bengio, Y. (2019). Compositional generalization in a deep seq2seq model by separating syntax and semantics. *arXiv:1904.09708 [cs, stat]*. arXiv: 1904.09708.

107. Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., and Lillicrap, T. (2016). Meta-learning with memory-augmented neural networks. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 1842–1850. JMLR.org.

108. Saxton, D., Grefenstette, E., Hill, F., and Kohli, P. (2019). Analysing mathematical reasoning abilities of neural models. In *International Conference on Learning Representations*.

109. Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117.

110. Schmidhuber, J., Zhao, J., and Wiering, M. (1996). Simple principles of metalearning. Technical report.

111. Schulz, E., Tenenbaum, J., Duvenaud, D. K., Speekenbrink, M., and Gershman, S. J. (2016). Probing the compositionality of intuitive functions. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

112. Shaw, P., Chang, M.-W., Pasupat, P., and Toutanova, K. (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.

113. Smolensky, P. (1988). The constituent structure of connectionist mental states: A reply to fodor and pylyshyn. *Southern Journal of Philosophy*, 26(S1):137–161.

114. Smolensky, P. (1990). Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, 46(1):159–216.

115. Snell, J., Swersky, K., and Zemel, R. (2017). Prototypical networks for few-shot learning. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

116. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

117. Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 3104–3112, Cambridge, MA, USA. MIT Press.

118. Tai, K. S., Socher, R., and Manning, C. D. (2015). Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing, China. Association for Computational Linguistics.

119. Thrun, S. and Pratt, L., editors (1998). *Learning to Learn*. Kluwer Academic Publishers, USA.

120. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

121. Vinyals, O., Blundell, C., Lillicrap, T., kavukcuoglu, k., and Wierstra, D. (2016). Matching networks for one shot learning. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

122. Wang, J. X., Kurth-Nelson, Z., Tirumala, D., Soyer, H., Leibo, J. Z., Munos, R., Blundell, C., Kumaran, D., and Botvinick, M. (2017). Learning to reinforcement learn. arXiv:1611.05763 [cs, stat].

123. Wayne, G., Hung, C.-C., Amos, D., Mirza, M., Ahuja, A., Grabska-Barwinska, A., Rae, J., Mirowski, P., Leibo, J. Z., Santoro, A., Gemici, M., Reynolds, M., Harley, T., Abramson, J., Mohamed, S., Rezende, D., Saxton, D., Cain, A., Hillier, C., Silver, D., Kavukcuoglu, K., Botvinick, M., Hassabis, D., and Lillicrap, T. (2018). Unsupervised Predictive Memory in a Goal-Directed Agent. arXiv:1803.10760 [cs, stat].

124. Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., Chi, E. H., Hashimoto, T., Vinyals, O., Liang, P., Dean, J., and Fedus, W. (2022a). Emergent abilities of large language models. *Transactions on Machine Learning Research*. Survey Certification.

125. Wei, J., Wang, X., Schuurmans, D., Bosma, M., brian ichter, Xia, F., Chi, E. H., Le, Q. V., and Zhou, D. (2022b). Chain of thought prompting elicits reasoning in large language models. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K., editors, *Advances in Neural Information Processing Systems*.

126. Winata, G. I., Cahyawijaya, S., Lin, Z., Liu, Z., Xu, P., and Fung, P. (2020). Meta-transfer learning for code-switched speech recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3770–3776, Online. Association for Computational Linguistics.

127. Xie, S. M., Raghunathan, A., Liang, P., and Ma, T. (2022). An Explanation of In-context Learning as Implicit Bayesian Inference. arXiv:2111.02080 [cs].

128. Yu, H., Zhang, H., and Xu, W. (2018). Interactive grounded language acquisition and generalization in a 2d world. In *International Conference on Learning Representations*.

129. Zhou, D., Schärli, N., Hou, L., Wei, J., Scales, N., Wang, X., Schuurmans, D., Cui, C., Bousquet, O., Le, Q. V., and Chi, E. H. (2023). Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*.