

# Equivariant Transduction through Invariant Alignment


Jennifer White<sup>δ</sup> Ryan Cotterell<sup>ι</sup>

<sup>δ</sup>University of Cambridge <sup>ι</sup>ETH Zürich

jw2088@cam.ac.uk ryan.cotterell@inf.ethz.ch

## Abstract

The ability to generalize compositionally is key to understanding the potentially infinite number of sentences that can be constructed in a human language from only a finite number of words. Investigating whether NLP models possess this ability has been a topic of interest: SCAN (Lake and Baroni, 2018) is one task specifically proposed to test for this property. Previous work has achieved impressive empirical results using a group-equivariant neural network that naturally encodes a useful inductive bias for SCAN (Gordon et al., 2020). Inspired by this, we introduce a novel group-equivariant architecture that incorporates a group-invariant hard alignment mechanism. We find that our network’s structure allows it to develop stronger equivariance properties than existing group-equivariant approaches. We additionally find that it outperforms previous group-equivariant networks empirically on the SCAN task. Our results suggest that integrating group-equivariance into a variety of neural architectures is a potentially fruitful avenue of research, and demonstrate the value of careful analysis of the theoretical properties of such architectures.

 <https://github.com/rycolab/equivariant-transduction>

## 1 Introduction

Humans painlessly process sentences they have never heard before. This feat is possible because they can construct the meaning of a sentence by composing the meaning of its parts. This phenomenon is known as **compositional generalization** in the computational linguistics literature (Lake and Baroni, 2018; Hupkes et al., 2020) and it is what enables the understanding of an infinite number of novel sentences from only a finite set of words (Chomsky, 1957; Montague, 1970). For example, without knowing what action the verb *to blink* describes, we know that *blink twice* indicates that this action should be performed two times. It is natural that we would like for neural network

walk right	→	RTURN WALK
walk and jump twice	→	WALK JUMP JUMP
jump left after walk	→	WALK LTURN JUMP

Figure 1: Examples of SCAN inputs and outputs

models of language to be endowed with this ability as well—and, indeed, if they are to achieve human-like performance, it is likely to be necessary.

There have been multiple proposals for methods of assessing these abilities in neural models (Bahdanau et al., 2019; Hupkes et al., 2020). One popular benchmark is the SCAN task (Lake and Baroni, 2018) and its derivatives (Ruis et al., 2020). SCAN involves translating natural language instructions into a sequence of actions executable by an agent navigating in an environment. Examples of these commands are shown in Figure 1. In order to test generalization capabilities, there are various test–train splits which deliberately hold out specific words. For example, the training set for the Add Jump split only contains *jump* in the simple input–output pair (*jump*, JUMP), while the test set includes commands where it is combined with modifiers, such as *jump twice* and *jump left*. Since the model has seen the effect of these modifiers on other verbs in training, a model with the ability to perform compositional generalization should be able to apply them to *jump*.

SCAN is an example of a task that could benefit from a group-equivariant network. When a network possesses the property of equivariance to a group  $G$ , acting on the input with an element of  $G$  results in the output differing by the action of the same group element. With an appropriate choice of group, this property can be used to imbue a network with the ability to generalize compositionally. If we have a set of input words, and a corresponding set of their output words, and we act on these with a permutation group which swaps words within these sets, then swapping one word for another in the input will result in the corresponding output words being swapped in the output. This effectively means that when the network has learned an example in

training it can generalize to any input which can be reached from the training example by acting on it with a group element. In the case of the SCAN task, the action on a sentence amounts to replacing a word with another vocabulary word from the same lexical class (e.g., replacing a verb with another verb). For example, if a  $G$ -equivariant network is trained on the SCAN task using a group  $G$  whose action amounts to swapping SCAN verbs, then the probability of run being mapped to RUN will be identical to the probability of jump being mapped to JUMP – thus even if only one verb is seen in training, it learns to apply observed patterns to the unseen verbs. This approach was first applied to SCAN by [Gordon et al. \(2020\)](#).

We present a novel group-equivariant architecture for the SCAN task which has a notable theoretical advantage over similar existing approaches: the effective orbit of the model is larger, which results in more robust generalization to novel examples. From one input command, our model can potentially generalize to an exponential number of unseen examples, where previous group-equivariant models could generalize only to a constant number. We also demonstrate the empirical effect of this advantage, showing that our model outperforms that of [Gordon et al.](#) across all splits of the SCAN task.

Concretely, we incorporate group equivariance into the hard alignment string-to-string transduction model described by [Wu et al. \(2018\)](#). Hard alignment differs from the more common soft attention ([Bahdanau et al., 2015](#)) in that each output symbol is aligned with precisely one input symbol, rather than calculating a weighting over all input symbols. Our model combines a group-invariant hard alignment mechanism with a group-equivariant transduction mechanism, which enables its improved generalization capabilities. Our findings motivate further exploration of group-equivariant architectures, and suggest that careful consideration of their provable equivariance properties is worthwhile.

## 2 Related Work

Many attempts have been made to quantify what it means for a model to exhibit compositional behaviour and to create models that can successfully generalize compositionally. [Hupkes et al. \(2020\)](#) provide a summary article on this topic, in which they describe different aspects of compositional generalization and formulate methods for assessing

a model on each aspect. In this work, we concern ourselves primarily with what [Hupkes et al.](#) term **systematicity**—the ability to understand an unseen combination of previously seen parts.

The SCAN task which we focus on has been approached in many ways. [Liu et al. \(2020\)](#) achieve state-of-the-art results on the task, achieving 100.0 across all splits, using a memory-augmented model, trained using reinforcement learning, which learns how to identify the symbolic functions described by specific phrases within inputs. [Lake \(2019\)](#) approach the task using meta sequence-to-sequence learning. Although this approach produces excellent results, it requires a bespoke meta-training approach for each generalization task. [Russin et al. \(2019\)](#) achieve good results by treating syntax and semantics separately. Their model separately calculates likely alignments between input and output words, and calculates how likely a word is to be produced conditioned on the input word with which it is aligned. This approach is similar in concept to the hard alignment model that we use as the base for our  $G$ -equivariant architecture.

Equivariant networks built using group convolutions ([Kondor and Trivedi, 2018](#)) have been used most in the field of computer vision, where they have been used to create Group Convolutional Neural Networks ( $G$ -CNNs) imbued with the property of invariance to transformations of an image such as rotation and reflection ([Cohen and Welling, 2016](#)). Recently these techniques have been adopted for NLP tasks: [Gordon et al. \(2020\)](#) applied group-equivariant networks to the task of compositional generalization. They used the building block of the group convolution to construct group-equivariant network components, including an LSTM, an attention mechanism and an embedding layer. Their model is a group-equivariant analogue of a standard LSTM-based sequence-to-sequence model. They evaluate it on the SCAN task and achieve high accuracies on the splits assessing systematicity. We build on their work by presenting an alternative group-equivariant architecture for this task, resulting in more robust generalization abilities and improved empirical performance.

## 3 Group-Equivariant Networks

### 3.1 Group Theory

Some understanding of the basics of Group Theory are helpful to understand group-equivariant networks. This section will briefly outline the key

concepts necessary for understanding our work.

**Definition 1.** A **group**  $(G, \circ)$  is a set  $G$  with a binary operation  $\circ$  with the following properties:

- (i) **Closure:** For any  $g, h \in G$ ,  $g \circ h \in G$ .
- (ii) **Identity:** There exists an element  $e$ , which we call the identity, such that for all  $g \in G$ ,  $e \circ g = g \circ e = g$ .
- (iii) **Inverse:** For any  $g \in G$  there is an element  $h \in G$  such that  $g \circ h = h \circ g = e$ . We call this element the inverse of  $g$  and we may write  $h = g^{-1}$ .
- (iv) **Associativity:** for any  $g_1, g_2, g_3 \in G$  we have  $(g_1 \circ g_2) \circ g_3 = g_1 \circ (g_2 \circ g_3)$ .

**Definition 2.** Let  $G$  be a group and  $X$  a set. A **left group action** of  $G$  on  $X$  is a function  $T : G \times X \rightarrow X$ . As shorthand, we write  $T(g, x) = g \circ x$ .

**Definition 3.** If a group  $G$  acts on a set  $X$  through a defined action, then the **orbit** of an element  $x \in X$  under  $G$  (which we write  $G \circ x$ ) is  $G \circ x = \{g \circ x \mid g \in G\}$ .

**Definition 4.** The **symmetric group**  $S_n$  is the group of all permutations of a set of  $n$  elements. The operation for this group is composition.

If  $G = S_n$  acts on a set  $X = \{x_1, \dots, x_n\}$ , then the canonical action of a group element permutes the items in the set. We will write elements of  $S_n$  in cycle notation, where  $g = (a_1 a_2 \dots a_m)$  indicates that  $g \circ a_i = a_{i+1}$  for  $i \in \{1, \dots, m-1\}$  with  $g \circ a_m = a_1$  (and by extension  $g \circ x_{a_i} = x_{g \circ a_i}$ ). Indices that do not appear in the cycle are left unchanged.  $()$  indicates the identity permutation, which maps each item to itself. An understanding of permutation groups will be key to our exposition, so we give an example of  $S_3$ .

**Example 1.**  $S_3$  is the symmetric group of permutations on 3 elements. In full,  $S_3 = \{(), (12), (13), (23), (123), (132)\}$ . It can act on any set with 3 elements. For example, if  $X = \{a, b, c\}$  then  $(12) \circ a = b$ ,  $(12) \circ b = a$  and  $(12) \circ c = c$ , since  $(12)$  maps the first element of the set to the second element and vice versa, while leaving the third element unchanged.

**Definition 5.** A group  $G$  is **cyclic** if there is an element  $g \in G$  such that all elements of  $G$  take the form  $g^i$ , where  $g^i = \underbrace{g \circ g \circ \dots \circ g}_i$  for some integer  $i$ . We say that  $G$  is generated by  $g$ , and write  $G = \langle g \rangle$ .

**Example 2.** One example of a cyclic group is a subgroup of  $S_n$  generated by the permutation  $g = (123 \dots p)$  for  $p \leq n$ . This permutation  $g$  has the effect of shifting each of these  $p$  elements by 1, so  $g^i$  has the effect of shifting each element by  $i$ , and  $g^p$  is equal to the identity permutation. We call this group a **cyclic shift group**.

To revisit the example of  $S_3$ ,  $G = \langle (123) \rangle = \{(), (123), (132)\}$  is an example of a cyclic shift group. If this group acts on a set of 3 elements, then the identity element  $()$ , leaves the items in the set unchanged, while  $(123)$  “shifts” each item along by one and  $(132)$  by two.

Cyclic shift groups will be useful in our work. If we have a set  $X$  of  $p$  objects that we wish to permute, the symmetric group of all possible permutations  $S_p$  contains  $p!$  elements. However, in our work we will consider each object in isolation, so we do not to consider every permutation: we only need a group  $G$  such that for a given  $x_1 \in X$ , for any element  $x_2 \in X$  there is a permutation  $g \in G$  such  $g \circ x_1 = x_2$ . This is true of the cyclic shift group  $G = \langle (123 \dots p) \rangle$ , which contains only  $p$  elements. By using cyclic shift groups in place of the symmetric groups, we are able to avoid unnecessarily slow calculations.

**Definition 6.** Let  $X$  and  $Y$  be sets and  $G$  be a group. Suppose the elements of  $G$  act on  $X$  and  $Y$  with actions denoted  $g \circ x$  and  $g \circ y$ ,  $x \in X$  and  $y \in Y$ . A function  $\alpha : X \rightarrow Y$  is **equivariant** with respect to  $G$  (or  $G$ -equivariant) if and only if  $\alpha(g \circ x) = g \circ \alpha(x)$  for all  $x \in X$  and  $g \in G$ . We say that  $\alpha$  is **invariant** to  $G$  if  $\alpha(g \circ x) = \alpha(x)$  for any  $x \in X$  and  $g \in G$ .

### 3.2 SCAN: A Test of Compositionality

The SCAN task was proposed by Lake and Baroni (2018) to test a model’s ability to generalize to unseen examples by composing known elements. The task involves translating between an instruction in a limited form of English with input alphabet  $\Sigma = \{\text{walk, jump, } \dots\}$  and an executable command with words taken from output alphabet  $\Delta = \{\text{WALK, JUMP, } \dots\}$ . As input the model receives a string  $\mathbf{x} \in \Sigma^*$ , such as `jump twice`, or `walk after run`, and as output it should produce a string  $\mathbf{y} \in \Delta^*$ , like `JUMP JUMP`, or `RUN WALK`.

SCAN contains several splits, each designed to test a different kind of generalization. We focus on the splits that aim to evaluate the systematicity of the model – that is to say how well it can under-

stand the unseen combination of previously seen parts. This is done by withholding certain combinations from the training set and then evaluating the model’s performance on these unseen combinations in the test set. For example, the training set of the Add Jump split only contains jump in the basic input–output pair (jump, JUMP), while the test set contains inputs where it is used in combination with other modifiers. Since these modifiers have been seen before with other verbs, this tests how well the model generalizes from what it has been exposed to in training.

### 3.3 Group-Equivariance in Transduction

We term a set of words whose function in SCAN is the same and that are used in identical contexts as a **lexical class**. We can see that, for example, any SCAN command containing walk would equally be a valid instruction if walk were replaced with jump, look, or run. We call this class the Verb lexical class. We divide the input vocabulary into lexical classes  $L_1^{\text{in}}, \dots, L_T^{\text{in}} \subseteq \Sigma$ . If we designate a lexical class to be our equivariant lexical class  $L_{\text{equi}}^{\text{in}}$ , we take the group  $G = \langle (12 \dots |L_{\text{equi}}^{\text{in}}|) \rangle$  to be the cyclic shift group of size  $|L_{\text{equi}}^{\text{in}}|$  – i.e., the group generated by a permutation that shifts each element along by one. For example, if  $L_{\text{equi}}^{\text{in}} = \{\text{walk}, \text{look}, \text{run}, \text{jump}\}$ , then  $G = \langle (1234) \rangle$ , whose elements are the permutations  $\{(), (1234), (13)(24), (1432)\}$ . The group acts on  $L_{\text{equi}}^{\text{in}}$  by permuting its elements, so  $(1234) \circ \text{walk} = \text{look}$ , for example. The group also acts in the same way on the output lexical class  $L_{\text{equi}}^{\text{out}} = \{\text{WALK}, \text{LOOK}, \text{RUN}, \text{JUMP}\}$ , with  $(1234) \circ \text{WALK} = \text{LOOK}$ . Previous work (Gordon et al., 2020) has defined the action of  $g \in G$  on a sentence  $\mathbf{x} = (x_1, \dots, x_N) \in \Sigma^*$  as  $g \circ \mathbf{x} = (g \circ x_1, \dots, g \circ x_N)$ .<sup>1</sup>

We now explain why group equivariance is a useful property for a task such as SCAN, and how it can lead to models that can generalize compositionally. Consider the SCAN task with input vocabulary  $\Sigma$ , output vocabulary  $\Delta$  and lexical classes  $L_{\text{equi}}^{\text{in}} \subseteq \Sigma, L_{\text{equi}}^{\text{out}} \subseteq \Delta$ . Let  $\xi : \Sigma^* \rightarrow \Delta^*$  be a transducer that constitutes a  $G$ -equivariant function for  $G = \langle (12 \dots |L_{\text{equi}}^{\text{in}}|) \rangle$ , with its action defined on  $\Sigma^*$  and  $\Delta^*$  as above. If we have an input–output pair  $(\mathbf{x}_0, \mathbf{y}_0) \in \Sigma^* \times \Delta^*$  such that  $\xi(\mathbf{x}_0) = \mathbf{y}_0$ , then for all input–output pairs  $(\mathbf{x}, \mathbf{y}) \in \Sigma^* \times \Delta^*$  such that  $(\mathbf{x}, \mathbf{y}) = (g \circ \mathbf{x}_0, g \circ \mathbf{y}_0)$  for some  $g \in G$ ,

<sup>1</sup>If  $x_n \in \Sigma \setminus L_{\text{equi}}^{\text{in}}, g \circ x_n = x_n$ .

we have  $\xi(\mathbf{x}) = \xi(g \circ \mathbf{x}_0) = g \circ \xi(\mathbf{x}_0) = g \circ \mathbf{y}_0 = \mathbf{y}$ . Plainly, this means that if our function successfully transduces a pair  $(\mathbf{x}_0, \mathbf{y}_0)$ , it will also transduce all pairs its orbit  $G \circ (\mathbf{x}_0, \mathbf{y}_0) = \{(g \circ \mathbf{x}_0, g \circ \mathbf{y}_0) \mid \forall g \in G\}$ . If we consider the case where  $\xi$  is obtained from the output of a neural network, this would mean that it could generalize compositionally to these examples even if the network had not been trained on them.

### 3.4 Constructing $G$ -Equivariant Networks

We now describe how to build a neural network that is equivariant to a group  $G$  using  $G$ -equivariant building blocks. We describe three such building blocks that we will use in the construction of our equivariant hard-alignment model.

①  **$G$ -Convolution.** The  $G$ -convolution is the extension of the standard convolution to an arbitrary finite group  $G$  (Kondor and Trivedi, 2018). Let  $G$  be a finite group and  $\mathbf{f} : \text{dom}(\mathbf{f}) \rightarrow \mathbb{R}^K$  an input function. Suppose we have  $D$  learnable filter functions where we denote the  $d^{\text{th}}$  filter function as  $\psi^{(d)} : \text{dom}(\mathbf{f}) \rightarrow \mathbb{R}^K$ . Then the  $G$ -convolution of  $\mathbf{f}$  with  $\{\psi^{(d)}\}_{d=1}^D$  is a  $|G| \times D$  matrix where each entry is given by the following:

$$\begin{aligned} G\text{-Conv}(\mathbf{f}; \psi)_{g,d} & \\ &= \sum_{h \in \text{dom}(\mathbf{f})} \mathbf{f}(h) \cdot \psi^{(d)}(g^{-1} \circ h) \end{aligned} \quad (1)$$

As shown in Cohen and Welling (2016, § 6.1),  $G$ -convolutions are  $G$ -equivariant.

②  **$G$ -Embed.** It is often desirable to embed input in a vector space – but for a  $G$ -equivariant network we require this step too to be equivariant. Gordon et al. (2020) show that a  $G$ -equivariant embedding can be obtained as a special case of a  $G$ -convolution where the input function for input word  $x$  is a one-hot encoding,  $x : \{1, \dots, |\Sigma|\} \rightarrow \{0, 1\}$ . Thus the learnable filter functions will be one-dimensional, and there is no longer a need for the sum, as it is zero at all but one value. We call our set of  $K$  filter functions  $\{\omega^{(k)}\}_{k=1}^K$ , with  $\omega^{(k)} : \{1, \dots, |\Sigma|\} \rightarrow \mathbb{R}$ . Then the  $G$ -equivariant embedding of an input  $x$  is a  $|G| \times K$  matrix  $e(x)$  with entries

$$\begin{aligned} e(x)_{g,k} &= G\text{-Embed}(x; \omega)_{g,k} \\ &= \omega^{(k)}(g^{-1} \circ x) \end{aligned} \quad (2)$$

③ ***G*-Decode.** The output  $\phi : G \rightarrow \mathbb{R}^D$  of some composition of equivariant layers based on *G*-convolutions will be a function over group elements, so in order to calculate logits over an output distribution of items  $\Delta$ , [Gordon et al. \(2020\)](#) use a decoding layer

$$G\text{-Dec}(\phi, \tilde{y}; \rho) = \sum_{h \in G} \phi(h) \cdot \rho(h^{-1} \circ \tilde{y}) \quad (3)$$

where  $\tilde{y} \in \Delta$  is a candidate output value and  $\rho : \{1, \dots, |\Delta|\} \rightarrow \mathbb{R}^D$  is a learnable filter function. This layer is not a form of group convolution, but instead is equivariant due to parameter-sharing ([Ravanbakhsh et al., 2017](#)).

## 4 An Equivariant Transducer

We now describe the architecture of our equivariant hard alignment transducer. Let  $\Sigma$  be an input alphabet and  $\Delta$  be an output alphabet. Given an input string  $\mathbf{x} \in \Sigma^*$  we create a model to calculate a probability distribution  $p(\mathbf{y} \mid \mathbf{x})$  over  $\mathbf{y} \in \Delta^*$ . We consider what it means for such a model to be equivariant with the following theorem.

**Theorem 1.** *Let  $p(\mathbf{y} \mid \mathbf{x})$  be a probability distribution over  $\Delta^*$ . If  $p(g \circ \mathbf{y} \mid g \circ \mathbf{x}) = p(\mathbf{y} \mid \mathbf{x})$  for any  $g \in G, \mathbf{x} \in \Sigma^*, \mathbf{y} \in \Delta^*$ , then the transducer  $\xi : \Sigma^* \rightarrow \Delta^*$  defined by  $\xi(\mathbf{x}) = \operatorname{argmax}_{\mathbf{y} \in \Delta^*} p(\mathbf{y} \mid \mathbf{x})$  is equivariant.*

*Proof.* Given an  $\mathbf{x}$ , let  $\mathbf{y}^* = \xi(\mathbf{x}) = \operatorname{argmax}_{\mathbf{y} \in \Delta^*} p(\mathbf{y} \mid \mathbf{x})$ . Now  $\xi(g \circ \mathbf{x}) = \operatorname{argmax}_{\mathbf{y} \in \Delta^*} p(\mathbf{y} \mid g \circ \mathbf{x})$ . But for any  $\mathbf{y} \in \Delta^*$ ,  $p(\mathbf{y} \mid g \circ \mathbf{x}) = p(g^{-1} \circ \mathbf{y} \mid g^{-1} \circ g \circ \mathbf{x}) = p(g^{-1} \circ \mathbf{y} \mid \mathbf{x})$ , so  $\xi(g \circ \mathbf{x}) = \operatorname{argmax}_{\mathbf{y} \in \Delta^*} p(g^{-1} \circ \mathbf{y} \mid \mathbf{x})$ . We know that the probability distribution conditioned on  $\mathbf{x}$  is maximized by  $\mathbf{y}^*$ , so we have  $g^{-1} \circ \xi(g \circ \mathbf{x}) = \mathbf{y}^*$  and thus  $\xi(g \circ \mathbf{x}) = g \circ \mathbf{y}^*$  and the transducer is equivariant.  $\square$

Moving forward we will use this equivalent definition to discuss the equivariance of our model.

### 4.1 A Hard-Alignment Transducer

As stated above, we aim to construct a distribution  $p(\mathbf{y} \mid \mathbf{x})$  where  $\mathbf{y} \in \Delta^*$  and  $\mathbf{x} \in \Sigma^*$ . The key idea behind a hard-alignment model is that the two strings  $\mathbf{x}$  and  $\mathbf{y}$  are aligned according to a latent alignment  $\mathbf{a} \in \mathcal{A}(\mathbf{x}, \mathbf{y})$ . Hard alignment is contrasted with a soft alignment ([Bahdanau et al., 2015](#); [Luong et al., 2015](#)), which does not have the interpretation as a latent variable. For  $\mathbf{x}$  of length  $N$ ,  $\mathbf{y}$  of length  $M$ , each  $\mathbf{a}$  is a vector in

$\{1, \dots, N\}^M$ . If we have  $a_m = n$ , then the input word  $x_n$  and output word  $y_m$  are aligned. Thus,  $\mathcal{A}(\mathbf{x}, \mathbf{y})$  consists of all non-monotonic alignments such that each word in  $\mathbf{y}$  aligns to exactly one word in  $\mathbf{x}$ . Because there is no annotation for alignments between the strings, we marginalize over alignments to compute the distribution:

$$\begin{aligned} p(\mathbf{y} \mid \mathbf{x}) &= \sum_{\mathbf{a} \in \mathcal{A}(\mathbf{x}, \mathbf{y})} p(\mathbf{y}, \mathbf{a} \mid \mathbf{x}) \quad (4) \\ &= \sum_{\mathbf{a} \in \mathcal{A}(\mathbf{x}, \mathbf{y})} \prod_{m=1}^M p(y_m \mid a_m, \mathbf{y}_{< m}, \mathbf{x}) p(a_m \mid \mathbf{y}_{< m}, \mathbf{x}) \end{aligned}$$

[Wu et al. \(2018\)](#) show that eq. (4) may be rewritten as follows using the distributive property

$$\prod_{m=1}^M \sum_{a_m=1}^N \underbrace{p(y_m \mid a_m, \mathbf{y}_{< m}, \mathbf{x})}_{\text{translator}} \underbrace{p(a_m \mid \mathbf{y}_{< m}, \mathbf{x})}_{\text{aligner}} \quad (5)$$

which is more efficient to compute. Specifically, this allows the distribution  $p(\mathbf{y} \mid \mathbf{x})$  to be computed  $\mathcal{O}(M \cdot N)$  rather than  $\mathcal{O}(|\mathcal{A}(\mathbf{x}, \mathbf{y})|)$ , which is exponential in both  $M$  and  $N$ .

This formulation allows us to completely separate the alignment probability  $p(a_m \mid \mathbf{y}_{< m}, \mathbf{x})$ , which calculates how likely an output word is to align with each input word, from the word translation probability  $p(y_m \mid a_m, \mathbf{y}_{< m}, \mathbf{x})$ , which calculates how likely each output word is to be produced when aligned with a given input word. This can also be viewed as separately treating the syntax (alignment probability) and semantics (word translation probability) of the input. In this sense, the model is similar to that of [Russin et al. \(2019\)](#).

We also experimented with a variation on this formulation in which the sum over  $a_m$  is replaced with taking the maximum, as well as an annealed variation. This is described in [App. A](#).

### 4.2 An Equivariant Translator

We now explain how the translator term in eq. (5) is defined in order to ensure group-equivariance. Because  $y_m$  and  $x_{a_m}$  are both individual words—as opposed to full sentences—the distribution  $p(y_m \mid x_{a_m}, a_m)$  is a simple classifier. We use a composition of the *G*-equivariant layers described in §3.4, so that

$$p(y_m \mid x_{a_m}, a_m) = \frac{\exp(G\text{-Dec}(\phi, y_m))}{\sum_{y' \in \Delta} \exp(G\text{-Dec}(\phi, y'))}$$

where  $\phi = G\text{-Conv}(e(x_{a_m}))$ . So, for each input word  $x_{a_m}$ , we first obtain a  $G$ -Embedding, then pass this through a  $G$ -Convolution, then use  $G$ -Decode to obtain logits over possible output words, which are then fed through a softmax to obtain probabilities.<sup>2</sup>

### 4.3 An Invariant Aligner

The alignment term is parameterized using a recurrent neural network. Before encoding, we replace each word with a symbol indicating its lexical class. Formally, if we have  $I$  disjoint lexical classes  $L_1^{\text{in}}, \dots, L_I^{\text{in}} \subseteq \Sigma$  that cover  $\Sigma$  and  $J$  disjoint lexical classes  $L_1^{\text{out}}, \dots, L_J^{\text{out}} \subseteq \Delta$  that cover  $\Delta$ , we define functions  $\ell_\Sigma : \Sigma \rightarrow \{1, \dots, I\}$  and  $\ell_\Delta : \Delta \rightarrow \{1, \dots, J\}$  such that  $\ell_\Sigma(w) = i$  iff  $w \in L_i^{\text{in}}$ ,  $\ell_\Delta(w) = j$  iff  $w \in L_j^{\text{out}}$ . We overload these so that for  $\mathbf{x} \in \Sigma^*$  of length  $N$ ,  $\ell_\Sigma(\mathbf{x}) = (\ell_\Sigma(x_1), \dots, \ell_\Sigma(x_N))$  and analogously  $\ell_\Delta(\mathbf{y}) = (\ell_\Delta(y_1), \dots, \ell_\Delta(y_M))$  for  $\mathbf{y} \in \Delta^*$  of length  $M$ . Then, our model is actually dependent on  $\ell_\Sigma(\mathbf{x})$  and  $\ell_\Delta(\mathbf{y}_{<m})$ . This has the effect of delexicalizing the input and equivalence-classing the words. For example, if the verb and direction lexical classes are being considered, the input walk left after run will, in effect, be represented as <verb> <direction> after <verb>. This substitution imbues the alignment model with some useful theoretical properties that will be discussed in §4.4.

The delexicalized input sequence  $\ell_\Sigma(\mathbf{x})$  of length  $N$  is encoded both forwards and backwards using an LSTM to produce hidden states  $\vec{\mathbf{h}}_n^{(\text{enc})}, \overleftarrow{\mathbf{h}}_n^{(\text{enc})} \in \mathbb{R}^{d_h}$  for  $n \in \{1, \dots, N\}$ . These are then concatenated to obtain  $\mathbf{h}_n^{(\text{enc})} = \vec{\mathbf{h}}_n^{(\text{enc})} \oplus \overleftarrow{\mathbf{h}}_n^{(\text{enc})} \in \mathbb{R}^{2d_h}$ . The output sequence  $\ell_\Delta(\mathbf{y})$  of length  $M$  is encoded forwards to produce  $\vec{\mathbf{h}}_m^{(\text{dec})} \in \mathbb{R}^{d_h}$  for  $m \in \{1, \dots, M\}$ . Then, we have the following distribution

$$p(a_m | \ell_\Delta(\mathbf{y}_{<m}), \ell_\Sigma(\mathbf{x})) = \frac{\exp(e_{ma_m})}{\sum_{n=1}^N \exp(e_{mn})} \quad (6)$$

where

$$e_{mn} = \vec{\mathbf{h}}_m^{(\text{dec})\top} \mathbf{T} \mathbf{h}_n^{(\text{enc})} \quad (7)$$

and  $\mathbf{T} \in \mathbb{R}^{d_h \times 2d_h}$  is a learned matrix.

### 4.4 Theoretical Results and Discussion

We know that each word-to-word translator  $p(y_m | x_{a_m}, a_m)$  is  $G$ -equivariant. We now explore in

<sup>2</sup>Additional non-linearities are introduced by passing the output of each layer through a tanh function.

more detail the theoretical properties of the entire model. We begin by considering the properties of the alignment model.

**Definition 7.** We say that a function  $f : X \times Y \rightarrow \mathbb{R}$  is invariant to a group  $G$  if for any  $x \in X, y \in Y, g \in G$ , we have  $f(g \circ x, g \circ y) = f(x, y)$ .

**Theorem 2.** The alignment model  $p(a_m | \ell_\Delta(\mathbf{y}_{<m}), \ell_\Sigma(\mathbf{x}))$  defined in §4.3 is invariant to  $G$ .

*Proof.* Let us take a group element  $g \in G$ . Then  $\ell_\Sigma(g \circ \mathbf{x}) = \ell_\Sigma(\mathbf{x})$ . We can see this because  $g \circ \mathbf{x} = (g \circ x_1, \dots, g \circ x_N)$  and  $g$  only permutes each word within its lexical class. Words within the same lexical class are mapped to the same value by  $\ell_\Sigma$ . A similar argument shows that  $\ell_\Delta(g \circ \mathbf{y}_{<m}) = \ell_\Delta(\mathbf{y}_{<m})$ . So  $p(a_m | \ell_\Delta(g \circ \mathbf{y}_{<m}), \ell_\Sigma(g \circ \mathbf{x})) = p(a_m | \ell_\Delta(\mathbf{y}_{<m}), \ell_\Sigma(\mathbf{x}))$  and the model is invariant.  $\square$

We can now examine the properties of the transducer as a whole.

**Theorem 3.** The model  $p(\mathbf{y} | \mathbf{x})$  defined in eq. (5) is  $G$ -equivariant.

*Proof.* We take a group element  $g \in G$ . Then

$$\begin{aligned} p(g \circ \mathbf{y} | g \circ \mathbf{x}) &= \prod_{m=1}^M \sum_{a_m=1}^N p(g \circ y_m | a_m, g \circ x_{a_m}) \\ &= p(a_m | \ell_\Delta(g \circ \mathbf{y}_{<m}), \ell_\Sigma(g \circ \mathbf{x})) \end{aligned}$$

We have seen that the translator part of the model is equivariant to  $G$ , and the alignment part is invariant, so we can rewrite the above as

$$\prod_{m=1}^M \sum_{a_m=1}^N p(y_m | a_m, x_{a_m}) p(a_m | \ell_\Delta(\mathbf{y}_{<m}), \ell_\Sigma(\mathbf{x}))$$

which is equal to  $p(\mathbf{y} | \mathbf{x})$  and thus the model is  $G$ -equivariant.  $\square$

We would like to understand what these equivariance properties actually mean for our model's ability to generalize to unseen sentences. To aid in the discussion of this question, we define the following.

**Definition 8.** Given a model  $p(\mathbf{y} | \mathbf{x})$  that is equivariant to a group  $G$  and a sentence pair  $(\mathbf{x}, \mathbf{y}) \in \Sigma^* \times \Delta^*$ , we define the **theoretical orbit** of  $(\mathbf{x}_0, \mathbf{y}_0)$  under the model as  $\Omega_T((\mathbf{x}_0, \mathbf{y}_0)) = \{(g \circ \mathbf{x}_0, g \circ \mathbf{y}_0) | g \in G\}$ . For any  $(x_i, y_i) \in \Omega_T((\mathbf{x}_0, \mathbf{y}_0))$ ,  $p(y_i | \mathbf{x}_i) = p(y_0 | \mathbf{x}_0)$ .

If we maximize  $p(\mathbf{y}_0 \mid \mathbf{x}_0)$  for a training pair  $(\mathbf{x}_0, \mathbf{y}_0)$ , any benefits of this are shared by elements of its theoretical orbit, allowing the model to generalize to these pairs. So the size of the theoretical orbit of a sentence pair is a way of quantifying how widely a model can generalize from that pair.

In the case of our model, with the equivariant lexical class corresponding to the 4 SCAN verbs, any sentence pair containing one of these verbs has a theoretical orbit of size 4. If  $(\mathbf{x}_0, \mathbf{y}_0)$  is a sentence pair containing two different SCAN verbs – for example, (walk after run, RUN WALK) – this means that its theoretical orbit contains 4 sentences out of the 16 sentences of this form that can be constructed. In addition to the theoretical orbit, which is guaranteed by the properties of the model, we also consider the behaviour of a trained model in practice.

**Definition 9.** Given a model  $p(\mathbf{y} \mid \mathbf{x})$  that is equivariant to a group  $G$  and a sentence pair  $(\mathbf{x}_0, \mathbf{y}_0) \in \Sigma^* \times \Delta^*$ , we define the **observed orbit** of  $(\mathbf{x}_0, \mathbf{y}_0)$  under the model as  $\Omega_O((\mathbf{x}_0, \mathbf{y}_0)) = \{(\mathbf{x}, \mathbf{y}) \in \Sigma^* \times \Delta^* \mid -\log p(\mathbf{y} \mid \mathbf{x}) = -\log p(\mathbf{y}_0 \mid \mathbf{x}_0)\}$ . Clearly  $\Omega_T((\mathbf{x}_0, \mathbf{y}_0)) \subseteq \Omega_O((\mathbf{x}_0, \mathbf{y}_0))$ .

To better understand how our model behaves, we examined the observed orbit of one of our best-performing models throughout training. We selected sentence pairs from SCAN containing 2 verbs, and generated all possible sentence pairs of the same form. For each form, there are 16 such pairs. While training our model, every five epochs we recorded the negative log-likelihood of each sentence pair as given by the model and counted how many of the 16 pairs had the same negative log-likelihood to find the size of the observed orbits.<sup>3</sup> We show the results for one sentence pair for the first 100 epochs of training in Figure 2. We can see that, initially, the 16 sentence pairs with the same form are in 4 distinct orbits of size 4, as predicted theoretically. However, as the model trains, all 16 sentences come to have the same negative log-likelihood, and this continues for the remaining duration of training.

To better understand why the observed orbit under our model is larger than predicted theoretically, we examine the effect of alignments in more depth. We consider a gold alignment for a SCAN sentence pair  $(\mathbf{x}, \mathbf{y})$  to be an alignment  $\mathbf{a}^*$  such that words from corresponding lexical classes are aligned and

<sup>3</sup>The values were considered to be equal if they were within floating point error, as checked by `torch.isclose()`.

each output word is aligned with an input word with the corresponding meaning (e.g., LTURN aligned with left, WALK aligned with walk).

Given  $\mathbf{x} \in \Sigma^N$ ,  $\mathbf{y} \in \Delta^M$  and an alignment  $\mathbf{a}$  between them, consider a group  $G^N$  with elements  $g = (g_1, \dots, g_N)$  with  $g_n \in G$ . Define its action on  $\mathbf{x}$  and  $\mathbf{y}$  by  $g \circ \mathbf{x} = (g_1 \circ x_1, \dots, g_N \circ x_N)$  and  $g \circ_{\mathbf{a}} \mathbf{y} = (g_{a_1} \circ y_1, \dots, g_{a_M} \circ y_M)$ . Note that the group  $G^N$  depends on the length of the string, and its action on  $\mathbf{y} \in \Delta^M$  depends on the alignment.

**Theorem 4.** Given  $\mathbf{x} \in \Sigma^N$ ,  $\mathbf{y} \in \Delta^M$  and a gold alignment  $\mathbf{a}^* \in \mathcal{A}(\mathbf{x}, \mathbf{y})$  between the two,  $p(g \circ_{\mathbf{a}^*} \mathbf{y} \mid g \circ \mathbf{x}, \mathbf{a}^*) = p(\mathbf{y} \mid \mathbf{x}, \mathbf{a}^*)$  for all  $g \in G^N$ , with actions defined as above.

*Proof.* For any  $g \in G^N$ ,  $p(g \circ_{\mathbf{a}^*} \mathbf{y} \mid g \circ \mathbf{x}, \mathbf{a}^*) = \prod_{m=1}^M p(g_{a_m^*} \circ y_m \mid g_{a_m^*} \circ x_{a_m^*}) = \prod_{m=1}^M p(y_m \mid x_{a_m^*}) = p(\mathbf{y} \mid \mathbf{x}, \mathbf{a}^*)$ .  $\square$

This means that for  $\mathbf{x}_0 \in \Sigma^N$ ,  $\mathbf{y}_0 \in \Delta^M$  and a gold alignment  $\mathbf{a}^*$ , the model is equivariant to  $G^N$  and thus the theoretical orbit of a sentence pair  $(\mathbf{x}_0, \mathbf{y}_0)$  under the model is  $|G|^N$  rather than  $|G|$  as it was without conditioning on the alignment.<sup>4</sup> Although this may seem irrelevant since we do not condition on gold alignments, we find that our trained models do approximate this. As training progresses, our model gradually becomes more confident about the correct alignment between the input and output. Once the model has confidently learned the gold alignment  $\mathbf{a}^*$  between an input–output pair  $\mathbf{x} \in \Sigma^*$ ,  $\mathbf{y} \in \Delta^*$ ,  $p(a_m \mid \mathbf{y}_{< m}, \mathbf{x})$  becomes very close to 0 for  $a_m \neq a_m^*$  and very close to 1 for  $a_m = a_m^*$ , meaning that  $p(\mathbf{y} \mid \mathbf{x})$  approaches  $p(\mathbf{y} \mid \mathbf{x}, \mathbf{a}^*) = \prod_{m=1}^M p(y_m \mid x_{a_m^*})$ . This means that for an input pair  $(\mathbf{x}_0, \mathbf{y}_0)$ , once the model has a high degree of confidence about the gold alignment  $\mathbf{a}^*$  and  $p(\mathbf{y}_0 \mid \mathbf{x}_0)$  approaches  $p(\mathbf{y}_0 \mid \mathbf{x}_0, \mathbf{a}^*)$ , the observed orbit of  $(\mathbf{x}_0, \mathbf{y}_0)$  will become this larger theoretical orbit. This explains what is shown in Figure 2.

## 5 Experiments and Results

Experiments were performed on the Simple, Add Jump, Around Right and Length splits of SCAN. In each case, models were trained on 90% of the train

<sup>4</sup>Unlike previous examples, theorem 4 only *directly* implies the  $G^N$ -equivariance of the restricted transducer  $\xi(\mathbf{x} \mid \mathbf{a}^*) = \operatorname{argmax}_{\mathbf{y} \in \Delta^M} p(\mathbf{y} \mid \mathbf{x}, \mathbf{a}^*)$ , since the action of  $G^N$  is only defined for  $\mathbf{y} \in \Delta^M$ . However, it does in fact imply the equivariance of the unrestricted transducer, since conditioning on the alignment means that  $p(\mathbf{y} \mid \mathbf{x}, \mathbf{a}^*) = 0$  for  $\mathbf{y} \in \Delta^Q$  if  $Q \neq M$ .

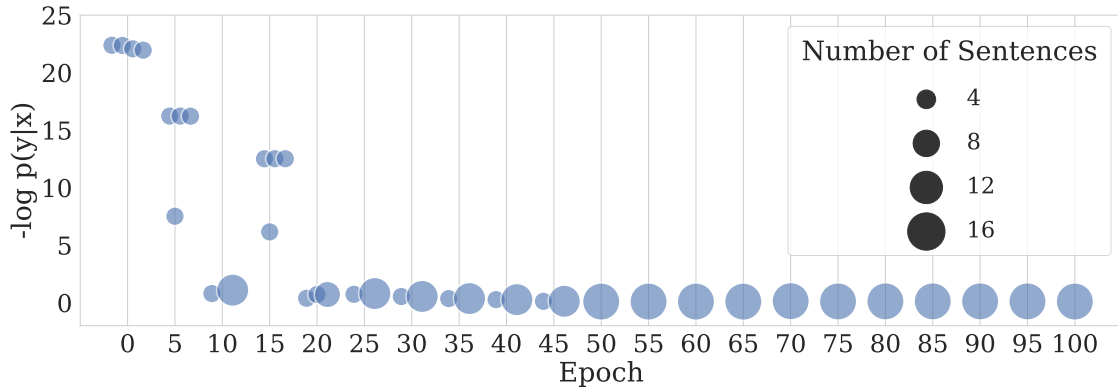


Figure 2: Observed orbits of sentence pairs  $(x, y)$  during the first 100 epochs of training, with  $x$  of the form  $\langle \text{verb}_1 \rangle$  right thrice after  $\langle \text{verb}_2 \rangle$  and  $y$  of the form  $\langle \text{VERB}_2 \rangle$  RTURN  $\langle \text{VERB}_1 \rangle$  RTURN  $\langle \text{VERB}_1 \rangle$  RTURN  $\langle \text{VERB}_1 \rangle$ .

Model	Simple	Add Jump	Around Right	Length
Russin et al. (2019)	100.0	91.0	28.9	15.2
Liu et al. (2020)	100.0	100.0	100.0	100.0
Gordon et al. (2020)	100.0	99.1	92.0	15.9
Equivariant Hard Alignment	100.0	100.0	100.0	28.5

Table 1: Accuracy achieved on SCAN task, presented alongside results from state-of-the-art systems

set, by minimizing the negative log-likelihood of observed sentence pairs, with 10% of the train set reserved to be used as a validation set. Models were selected to have the lowest loss on the validation set, and then evaluated on the test set.

For the Add Jump split, the group used was the cyclic shift group of size 4 acting on the set of SCAN verbs. For the Around Right split, which withholds the combination around right, the group used was the cyclic shift group of size 2 acting on the set of directions. Hyperparameters were selected through a random hyperparameter search. At test time, outputs were decoded using a beam search with 3 beams.

Results are shown in Table 1. We can see that our model outperforms Gordon et al.’s (2020) similar group-equivariant model on all splits, achieving 100% accuracy on all but the Length split, demonstrating empirically the advantage of the wider generalization capabilities that we explained theoretically. It still falls short of Liu et al.’s (2020) state-of-the-art memory-augmented model on the Length split. Since the equivariance of our model primarily targets systematicity, which is not the ability targeted by the Length split, it is not surprising that it does not perform at state-of-the-art levels in this split. We note, however, that our model does sub-

stantially outperform both Gordon et al.’s (2020) group-equivariant model, and Russin et al.’s (2019) semantic-syntactic alignment model on this split, which are the two models most similar to our own.

## 6 Limitations and Future Work

Group-equivariant networks have not yet been widely adopted in NLP, and we feel that there is potential for more tasks to be identified that could benefit from the application of group equivariance. A further line of investigation would be to develop and assess group-equivariant versions of a wider variety of architectures. The model used by Gordon et al. (2020) is an equivariant version of a standard LSTM-based sequence-to-sequence model, and the model used in our work is based on the hard alignment model used by Wu et al. (2018) – these are only two of many possible architectures. Variants of other architectures may possess different equivariance properties from both of these models, and may perform well on different tasks. Future work could incorporate group equivariance into more architectures and assess their theoretical properties and empirical performance.

A key limitation of group-equivariant networks is the need to understand and specify the group to which it is equivariant. This makes it difficult



to apply these networks to real world tasks rather than artificial datasets such as SCAN. Future work could investigate ways to identify from data which words should be in a lexical class, or ways to allow group-equivariant networks to deal with open vocabulary problems. Some work has been done on learning input and output vocabulary alignments in the context of SCAN (Akyurek and Andreas, 2021), so it is possible that this could be used to improve group-equivariant architectures by reducing or eliminating their reliance on lexical classes and groups known *a priori*.

## 7 Conclusion

In this work, we proposed and implemented a string-to-string transduction model which combines a group-invariant sum over hard alignments and a group-equivariant output word probability to create a model which is equivariant to the swapping of words in the same lexical class. We applied this model to the SCAN task, finding that it is successful in allowing the model to generalize compositionally. We show theoretically that our model’s structure allows for wider generalization to novel sentences than existing group-equivariant approaches and demonstrate this empirically. We suggest that this is strong motivation to explore group-equivariant variants of other architectures, and to investigate other tasks which may benefit from group-equivariant models, as well as suggesting that theoretical analysis of equivariance properties may be a useful tool in understanding performance differences.

## References

- Ekin Akyurek and Jacob Andreas. 2021. [Lexicon learning for few shot sequence modeling](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4934–4946, Online. Association for Computational Linguistics.
- Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. 2015. [Neural machine translation by jointly learning to align and translate](#). In *International Conference on Learning Representations*.
- Dzmitry Bahdanau, Harm de Vries, Timothy J O’Donnell, Shikhar Murty, Philippe Beaudoin, Yoshua Bengio, and Aaron Courville. 2019. [Closure: Assessing systematic generalization of clevr models](#). *arXiv preprint arXiv:1912.05783*.
- Noam Chomsky. 1957. *Syntactic Structures*. De Gruyter Mouton.
- Taco Cohen and Max Welling. 2016. [Group equivariant convolutional networks](#). In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2990–2999, New York, New York, USA. PMLR.
- Jonathan Gordon, David Lopez-Paz, Marco Baroni, and Diane Bouchacourt. 2020. [Permutation equivariant models for compositional generalization in language](#). In *International Conference on Learning Representations*.
- Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. 2020. [Compositionality decomposed: How do neural networks generalise?](#) In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 5065–5069. International Joint Conferences on Artificial Intelligence Organization.
- Diederik P. Kingma and Jimmy Ba. 2015. [Adam: A method for stochastic optimization](#). In *International Conference on Learning Representations*.
- Risi Kondor and Shubhendu Trivedi. 2018. [On the generalization of equivariance and convolution in neural networks to the action of compact groups](#). In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2747–2755. PMLR.
- Brenden Lake and Marco Baroni. 2018. [Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks](#). In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2873–2882. PMLR.
- Brenden M. Lake. 2019. [Compositional generalization through meta sequence-to-sequence learning](#). In *Advances in Neural Information Processing Systems*.
- Qian Liu, Shengnan An, Jian-Guang Lou, Bei Chen, Zeqi Lin, Yan Gao, Bin Zhou, Nanning Zheng, and Dongmei Zhang. 2020. [Compositional generalization by learning analytical expressions](#). In *Advances in Neural Information Processing Systems*, volume 33, pages 11416–11427. Curran Associates, Inc.
- Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. [Effective approaches to attention-based neural machine translation](#). In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.
- Richard Montague. 1970. [Universal grammar](#). *Theoria*, 36(3):373–398.

- Siamak Ravanbakhsh, Jeff Schneider, and Barnabás Póczos. 2017. [Equivariance through parameter-sharing](#). In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2892–2901. PMLR.
- Laura Ruis, Jacob Andreas, Marco Baroni, Diane Bouchacourt, and Brenden M Lake. 2020. [A benchmark for systematic generalization in grounded language understanding](#). In *Advances in Neural Information Processing Systems*, volume 33, pages 19861–19872.
- Jake Russin, Jason Jo, Randall C. O’Reilly, and Yoshua Bengio. 2019. [Compositional generalization in a deep seq2seq model by separating syntax and semantics](#). *arXiv preprint arXiv:1904.09708*.
- Shijie Wu, Pamela Shapiro, and Ryan Cotterell. 2018. [Hard non-monotonic attention for character-level transduction](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4425–4438, Brussels, Belgium. Association for Computational Linguistics.

## A Model Variants

In addition to the sum-based model presented in the main body of the paper, we also experimented on some variations of the model, which we explain here.

### A.1 Max Model

The max version of the model is given by

$$p(\mathbf{y} \mid \mathbf{x}) = \prod_{m=1}^M \max_{1 \leq a_m \leq N} p(y_m \mid a_m, \mathbf{y}_{< m}, \mathbf{x}) p(a_m \mid \mathbf{y}_{< m}, \mathbf{x}) \quad (8)$$

where all terms are the same as for the sum model.

### A.2 Annealed Max Model

This model proved difficult to train, so we also conducted experiments with an annealed max model, which is given by

$$p(\mathbf{y} \mid \mathbf{x}) = \prod_{m=1}^M \sum_{a_m=1}^N \alpha_{a_m}^m p(y_m \mid a_m, \mathbf{y}_{< m}, \mathbf{x}) p(a_m \mid \mathbf{y}_{< m}, \mathbf{x}) \quad (9)$$

where  $\alpha^m$  is given by

$$\alpha_{a_m}^m = \frac{\exp\left(\frac{1}{\tau} p(y_m \mid a_m, \mathbf{y}_{< m}, \mathbf{x}) p(a_m \mid \mathbf{y}_{< m}, \mathbf{x})\right)}{\sum_{a'_m=1}^N \left(\frac{1}{\tau} p(y_m \mid a'_m, \mathbf{y}_{< m}, \mathbf{x}) p(a'_m \mid \mathbf{y}_{< m}, \mathbf{x})\right)} \quad (10)$$

where  $0 < \tau \leq 1$  is the temperature. As  $\tau$  approaches 0,  $\alpha^m$  approaches a one-hot vector indicating the argmax of  $p(y_m \mid a_m, \mathbf{y}_{< m}, \mathbf{x}) p(a_m \mid \mathbf{y}_{< m}, \mathbf{x})$ , and thus eq. (9) becomes close to eq. (8). During training,  $\tau$  is gradually decreased, so that the fully trained model is, in effect, a max model. The starting value of  $\tau$ , as well as the schedule on which it is decreased, are hyperparameters.

### A.3 Model Comparison

In Table 2 we show the best accuracy achieved on each split by each variant of our model.

Model	Simple	Add Jump	Around Right	Length
Sum	100.0	100.0	100.0	28.5
Max	100.0	99.9	99.9	18.3
Annealed Max	99.9	99.9	99.9	19.7

Table 2: Accuracy achieved on SCAN task by each variant of our model

## B Reproducibility

The model was optimized using Adam (Kingma and Ba, 2015) with a learning rate of 0.001. We chose hyperparameters through a random search. Each hyperparameter included in the search is described in Table 3 along with the range that was searched. For each split, sets of hyperparameters were randomly sampled and models with those hyperparameters were trained and evaluated. Table 4 shows the value of the hyperparameters for the best-achieving model on each split. In Table 5 and Table 6 we also include the best-performing hyperparameters for the max and annealed variations of the model.

Hyperparameter	Details	Range
Dimension of $G$ -Embedding	Size of $G$ -Embedding layer	5-256
Number of Filters	Number of filters used in $G$ -Convolution layer	5-256
Embedding Dimension	Dimension of embedding layer in alignment model	5-256
Hidden Size	Size of LSTM layers used in alignment model	5-256
Batch Size	-	8 - 64

Table 3: Hyperparameter ranges that were searched

Split	Dimension of $G$ -Embedding	Number of filters	Embedding Dimension	Hidden Size	Batch Size
Simple	20	24	36	6	8
Simple	6	13	67	13	8
Add Jump	122	7	223	67	8
Around Right	100	7	122	36	8
Around Right	20	20	9	55	8
Around Right	45	182	100	9	8
Length	45	24	11	149	32

Table 4: Values of hyperparameters for the best-performing sum-model on each split tested. Where more than one set of hyperparameters is given for a split, both performed equally well.

Split	Dimension of $G$ -Embedding	Number of filters	Embedding Dimension	Hidden Size	Batch Size
Simple	16	24	182	36	8
Add Jump	30	11	182	55	16
Around Right	30	223	122	11	8
Length	36	55	30	4	16

Table 5: Values of hyperparameters for the best-performing max-model on each split tested.

Split	Dimension of $G$ -Embedding	Number of filters	Embedding Dimension	Hidden Size	Batch Size
Simple	182	122	223	11	8
Add Jump	122	7	223	67	8
Around Right	55	100	45	30	8
Length	13	9	11	16	16

Table 6: Values of hyperparameters for the best-performing annealed max-model on each split tested.

Training Data Percentage	Sum Model	Max Model	Annealed Model	Gordon et al.
1	42.14	0.47	33.26	<b>45.75</b>
2	68.09	56.28	58.36	<b>73.53</b>
4	<b>89.59</b>	9.2	81.22	89.54
8	94.67	<b>99.23</b>	95.61	96.60
16	98.88	<b>99.48</b>	98.44	97.12
32	<b>99.86</b>	99.48	97.41	97.47
64	99.67	<b>99.88</b>	99.27	99.67

Table 7: Accuracy obtained by each model in low-data conditions

## C Low-Data Experiments

To test our model’s ability to learn from small amounts of training data, we trained and evaluated a model with the best-performing hyperparameters on the Simple split using the low-data splits of SCAN. These splits have training sets containing 1%, 2%, 4%, 8%, 16%, 32% and 64% of the total examples in the Simple split. For comparison, we repeated this for all variants of our model, as well as for the equivariant model used by Gordon et al. (2020). The results are shown in Table 7. We can see that in the lowest data conditions, our model doesn’t generalize as well as Gordon et al.’s (2020) model. For the sets containing 16% or more of the data, our model performs better. We theorize that this is due to difficulty learning a high-quality alignment in the lowest-data conditions. These results also show the volatility of the Max variant of our model – it is able to reach above 99% accuracy with less data than any of the other models, but with any less data it struggles to learn much at all.

## D SCAN Lexicon

For further context we provide lists of the full SCAN lexicon for input and output in Table 8 and Table 9 respectively. In Table 10 we list the words in the Direction lexical class in the input and output lexicon. In Table 11 we do the same for the Verbs lexical class. All other input words were in single-item lexical classes.

Input Words
run
walk
look
jump
left
right
after
and
turn
around
twice
thrice
opposite
around

Table 8: All input vocabulary for the SCAN task

Output Words
RUN
WALK
LOOK
JUMP
LTURN
RTURN

Table 9: All output vocabulary for the SCAN task

Input	Output
right	RTURN
left	LTURN

Table 10: Direction lexical class in input and output vocabulary

Input	Output
run	RUN
walk	WALK
look	LOOK
jump	JUMP

Table 11: Verb lexical class in input and output vocabulary