NEURAL NETWORKS CAN UNDERSTAND COMPOSITIONAL FUNCTIONS THAT HUMANS DO NOT, IN THE CONTEXT OF EMERGENT COMMUNICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

We show that it is possible to craft transformations that, applied to compositional grammars, result in grammars that neural networks can learn easily, but humans do not. This could explain the disconnect between current metrics of compositionality, that are arguably human-centric, and the ability of neural networks to generalize to unseen examples. We propose to use the transformations as a benchmark, ICY, which could be used to measure aspects of the compositional inductive bias of networks, and to search for networks with similar compositional inductive biases to humans. As an example of this approach, we propose a hierarchical model, HU-RNN, which shows an inductive bias towards position-independent, word-like groups of tokens.

1 INTRODUCTION

Statistical association language models produce impressive results in domains such as summarization, and few-shot learning (e.g. Zhang et al. (2020), or Brown et al. (2020)). However, it is unclear to what extent such tasks require creative invention by the neural models. Thus, we target a slightly different task of 'emergent communication'. Tabula rasa agents placed in a collaborative scenario emerge their own communicative code (e.g. Lazaridou et al. (2018) and Foerster et al. (2016)). We wish to reproduce aspects of the development of human natural language (e.g. Pinker & Bloom (1990), Berwick et al. (2012)). A key aspect is compositionality: the meaning of an utterance is a function of the meaning of the parts. Agents in emergent communication scenarios empirically do not naturally produce compositional output, as measured by human evaluation, and by compositional metrics, e.g. Kottur et al. (2017).

Kirby et al. (2008) showed in human experiments that artificial languages evolved to become more compositional when transmitted from one human to another. However, in the case of artificial models, Griffiths & Kalish (2007) showed that for a broad range of conditions, transmission of languages across generations converges to the prior. For artificial models, a key question thus is: what are the priors? To what extent do commonly used models incorporate a compositional inductive bias?

To go further, we need a concrete definition of compositionality. We use the definition of compositionality from Andreas (2019): an utterance representing the combined meaning of two sub-utterances should be a deterministic function $g(\cdot, \cdot)$ of the two sub-utterances. This is a broad definition of compositionality, and includes holistic mappings, which do not generalize. We thus consider two subsets of compositionality, which we term 'generalizable' compositionality, and 'human' compositionality. Human compositionality is defined to be compositional functions which can be used by humans. Generalizable composition is defined to be any composition function which allows generalization. Figure 1 depicts these subsets of composition space, as well as a subset 'neural', depicting composition functions usable by current neural models.

Our current metrics of composition implicitly target human compositionality. We hypothesize that a consistently observed disconnect between the measured compositionality of emergent communication grammars, and their ability to generalize (Chaabouni et al., 2020), is a direct consequence of our metrics of compositionality targeting human compositionality.

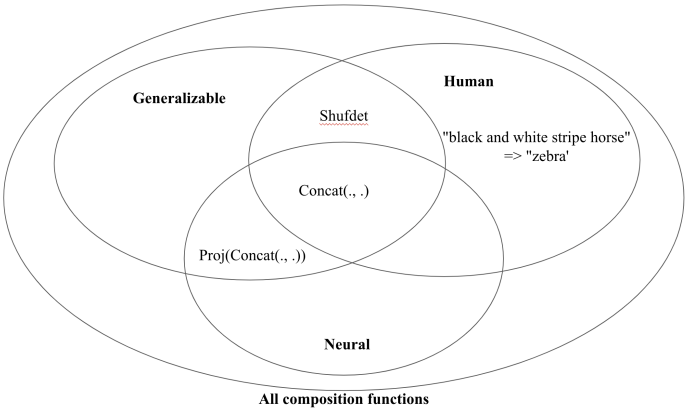

Figure 1: Subsets of composition space

We present specific examples of generalizable composition functions, which neural models can acquire easily, but which humans do not recognize as compositional, and which current compositional metrics consider to be non-compositional. In addition, we present a grammar, SHUFDET, whose composition humans can understand but which neural models cannot. We propose a novel neural architecture, HU-RNN, that can acquire SHUFDET faster than other neural models.

What we can learn from this is three-fold. Firstly, when we talk about compositionality, we should be clear about whether we mean human compositionality, generalizable compositionality, or some other kind of compositionality. Secondly, we should be clear about what our goal is when we wish for emergent communication games to emerge compositional language. Is our goal to make the language appear compositional to humans, or simply that the language appear compositional to neural networks? Thirdly the compositional inductive bias of current neural networks is quite different from that of humans. There are generalizable compositions that neural networks can use that humans cannot; and similarly there are compositional functions, e.g SHUFDET, that humans can use that current neural networks do not.

Our contributions are:

- demonstrate transformations, which we can apply to concatenation grammars which give rise to grammars whose compositional structure:
  - appears to current metrics of compositionality as non-compositional
  - is opaque to humans
  - does not affect acquisition speed of neural models
- we measure the performance of these transformations:
  - using current compositional metrics
  - using human evaluation
  - using a selection of standard neural models
- in addition we propose a transformation, SHUFDET, which we show that humans can readily understand, but which neural models acquire slowly
- as an example of using our transformations to search for models with a compositional inductive bias more aligned with that of humans, we propose a model, HU-RNN, that shows faster acquisition speed for SHUFDET

2 BACKGROUND

2.1 GENERAL FRAMEWORK

We assume a signaling game (Lewis, 2008). A Sender receives an object o, and generates a message m, Figure 2. A Receiver receives the message m, and decodes the message into a prediction $\hat{o}$

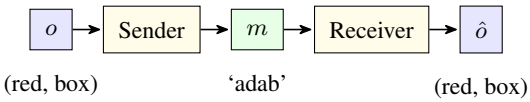

Figure 2: Signaling Game. 'adab' is an example message. (red, box) is an example object.

of the original object o. The message m is a fixed length utterance of c_{len} symbols drawn from a vocabulary V of size $|V|$. Each object o comprises n_{att} attributes, $\{o^{(1)}, \ldots, o^{(n_{att})}\}$ each with n_{val} possible values. We draw the attributes $o^{(j)}$ from a vocabulary Σ, of size $|\Sigma| = n_{att} \cdot n_{val}$, where $\Sigma_{j,*}$ are possible values for attribute $o^{(j)}$. For example, $\Sigma_{1,*}$ could represent color; $|\Sigma_{1,*}| = n_{val}$ would be the number of possible colors; and $\Sigma_{1,3}$ could mean 'red'. When presented to a neural network, o is represented as the concatenation of n_{att} one-hot vectors, each of length n_{val}.

In emergent communication games, we can ssign a reward $r = 1$ if $\hat{o} = o$, and train using REINFORCE (Williams, 1992). The agents co-ordinate to form a language $\mathcal{G}$ comprising pairs of objects and messages $\mathcal{G} = \{(o_1, m_1), \ldots, (o_N, m_N)\}$, where $N = n_{val}^{n_{att}}$ is the number of objects in the object space $\mathcal{O}$ (Lazaridou et al., 2018).

In our work, we will consider the Sender or Receiver models in isolation, and attempt to obtain insights into their intrinsic compositional inductive biases.

2.2 Compositionality metrics

To measure compositionality, Andreas (2019) proposed TRE. TRE is a mathematical implementation of a definition of compositionality that the whole is a composition of the parts. TRE imposes no constraints on the composition function. Practical implementations of TRE provide opinions on allowed composition function. Section 7 of Andreas (2019) (hereafter 'TRE7') takes the composition function to be the concatenation of sub-messages, followed by parameterized permutation. Chaabouni et al. (2020)'s posdis assumes a message whose length equals the number of attributes in the input object, and where each message token, in a specific position, represents a single attribute. Their bosdis constrains the meaning of a token to be invariant with position. Thus, these metrics assume that we can partition messages into groups of one or more message tokens that each represent one attribute. Resnick et al. (2020) indeed explicitly incorporate such a partition function into their resent metric. Lastly, Brighton & Kirby (2006) proposed topsim ('topological similarity'), which is a mature, widely-used metric, with few assumptions. topsim reports the correlation between the distances between objects, and distances between messages, over pairs of (object, message) tuples. The distance between messages is typically taken to be the L1 norm, or an edit distance. topsim will be a maximum when groups of message tokens map to individual attributes, and are combined with concatenation, possibly followed by permutation, similar to TRE7. All assume a permutation over concatenation as the composition function.

We will see in our experiments that it is possible to apply simple transforms to messages, which do not affect much the acquisition speed of neural models. However, which render the message apparently non-compositional to humans, and to our current metrics of compositionality.

2.3 Other work on compositionality

One approach to investigating the compositional inductive biases of models is to run many emergent communication experiments. This is time-consuming, noisy, and entangles many factors of variation. Importantly, it is unclear how to inspect the compositional characteristics of the resulting languages. We choose an alternative approach of generating languages which exhibit specific deviations from a perfectly compositional language; and measuring how easily each model can fit these artificial languages. Our approach is similar to that used in Li & Bowling (2019), Kharitonov & Baroni (2020) and Resnick et al. (2020). However, Li & Bowling (2019) only considers a single transformation (permutation); Kharitonov & Baroni (2020) focus on the effect of compositionality on generalization; and Resnick et al. (2020) investigates primarily the effect of capacity. Hupkes et al. (2020) and White & Cotterell (2021) use artificially created languages to test neural model's understanding of compositional forms that appear in natural language. In our work, we search for languages which models can fit to easily, but which a human might consider non-compositional.

3 METHODOLOGY

In our work, we will train the Sender or Receiver in isolation, using artificial languages of our choosing. We seek grammars which score poorly on compositional metrics, appear non-compositional to humans, but which demonstrate a fast acquisition speed by neural networks. The general approach we follow is to start with concatenation grammars, and apply transformations to the linguistic representations which we hope might not affect the compositional form, as perceived by neural networks.

3.1 ARTIFICIAL GRAMMARS

3.1.1 CONCATENATION GRAMMAR (CONCAT)

We start from a simple concatenation composition. We sample a bijective map from each $\Sigma_{i,j}$ to sub-messages $w_{i,j}$, of length c_w, drawn from vocabulary V, where $c_w = c_{len}/n_{att}$. Given an object o, we map each attribute value $o^{(j)}$ to a sub-message $w_{j,o^{(j)}}$ (i.e. the word for attribute j and attribute value $o^{(j)}$), and concatenate the sub-messages. For example, attribute value 'red' could map to subsequence 'adaa', and 'box' could map to sub-message 'ccad'. Thus object (red, box) would map to message 'adaaccad', and any red object would have a message starting with 'adaa'.

3.1.2 HOLISTIC GRAMMAR (HOL)

For each object o_n we generate a random message m_n. This provides a baseline to compare the acquisition speed on other grammars against.

3.1.3 PERMUTED GRAMMAR (PERM)

We sample a single permutation, and apply this to all messages in a sampled concatenation grammar $\mathcal{G}_{concat}$, to form a permuted language $\mathcal{G}_{perm}$.

3.1.4 RANDOM PROJECTION GRAMMAR (PROJ)

Neural networks apply projections at each layer. We hypothesize therefore that the ground truth output given to a neural network can be arbitrarily projected, without affecting the acquisition speed. Let use first consider the general, non-discrete, case, given dataset $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_N, \mathbf{y}_N)\}$, where each $(\mathbf{x}_n, \mathbf{y}_n)$ is a pair of input and output vectors. We hypothesize that we can apply any non-singular projection matrix P to all y_n, forming $y'_n = Py_n$, without affecting the acquisition speed of a neural network.

In the case of a discrete message m_n, we first expand to one-hot, vectorize, then apply. We form a one-hot matrix $m_n \rightarrow m_n^{onehot} \in \mathbb{R}^{c_{len} \times |V|}$, adding a new dimension over V. We vectorize to form $\text{vec}(m_n^{onehot}) \in \mathbb{R}^{(c_{len}|V|)}$, then apply a projection $P \in \mathbb{R}^{(c_{len}|V|) \times (c_{len}|V|)}$. After unvectorizing and taking the argmax to recover a new discrete message, we obtain:

$$m_n^{proj} = \arg\max_V \text{vec}^{-1}(P\text{vec}(m_n^{onehot}))$$

We sample a single projection matrix P per generated language. To the best of our knowledge, there is no equivalent composition operator to PROJ in natural language.

3.1.5 CUMULATIVE ROTATION GRAMMAR (ROT)

Vanilla recurrent neural networks (RNNs) take the input from the previous time-step and project it. Consider a transformation where we add the transformed output from the previous timestep to the current timestep:

$$m_n^{(j,rot)} = (m_n^{(j-1,rot)} + m_n^{(j)}) \mod |V|$$

(where $m^{(j)}$ is the message symbol at position j, and $m^{(j,rot)}$ is the message symbol at position j in the cumulatively rotated message. $\mod$ is the modulo operator).

We hypothesize that such a transformation is aligned with the transformations in a vanilla RNN, and so might be acquired quickly. Meanwhile, ROT has no equivalent composition function in human natural language.

3.1.6 RELOCATABLE ATOMIC GROUPS OF TOKENS (SHUFDET)

We would like to encourage the models to emerge relocatable atomic groups of tokens, that is something similar to words in natural language. We want a deterministic shuffling, so that the Sender model knows which variation to output. In natural language, some word orders are dependent on the values of certain words. For example, in French, the adjective 'neuve' follows a noun, whereas 'nouvelle' precedes it. Thus we use the value of the last attribute of the meaning to determine the order of the sub-messages w, prior to concatenation. That is, for each possible value of the last attribute, we sample a permutation, and we apply this same permutation to all messages having the same last attribute value.

SHUFDET contrasts with the other artificial grammars we propose in that we feel that models with a similar compositional inductive bias to humans should acquire these grammars quickly. In Appendix H we present an additional variation SHUF.

3.2 COMPOSITIONALITY METRICS

In addition to measuring model acquisition speed, we evaluate samples of each of the grammars for compositional metrics: bosdis, posdis, TRE7 and topsim. Since we have $c_{len} > n_{att}$, we violate assumptions of bosdis and posdis. However, we provide their scores for completeness. We wanted to use in addition resent. However, minimizing over all possible message partitions took combinatorial time. Therefore we relaxed the minimization, to give a new metric HCE, which we describe in Appendix D.

3.3 NEURAL MODELS UNDER TEST

We primarily target neural models frequently used in emergent communication and natural language processing. In addition we experiment with the evolved Sender model from Dagan et al. (2020), an RNN decoder with zero'd inputs, and a novel architecture, HU-RNN.

3.3.1 RNN DECODER WITH ZERO'D INPUTS (RNNZERO)

An RNN comprises an inner cell $o_t, h_t = \text{RNN}(x_t, h_{t-1})$, where o_t is output at time step t, h_t is hidden state, and x_t is input. When used as a decoder, the output is fed back auto-regressively: $o_t, h_t = \text{RNN}(W_{hi}o_{t-1}, h_{t-1})$, where W_{hi} is a projection. We experiment in addition with a decoder where the input at each time step is all zeros: $o_t, h_t = \text{RNN}(\mathbf{0}, h_{t-1})$. We use a '-Z' suffix to denote this, e.g. 'LSTM-Z', when using an LSTM-based decoder (Hochreiter & Schmidhuber, 1997).

In many frameworks, e.g. PyTorch (Paszke et al., 2019), RNN-Zs uses fewer lines of code, and arguably have lower Kolmogorov complexity (Kolmogorov, 1963). We show that their compositional inductive bias is sometimes better than the auto-regressive variant.

3.3.2 HIERARCHICAL UNIT RNN (HU-RNN)

Hierarchical-Unit RNNS ('HU-RNNs') are fully differentiable, and might encourage an inductive bias towards receiving and sending atomic relocatable groups of tokens, i.e. for SHUFDET.

'HUSendZ' is a Sender model. There are two recurrent neural network layers ('RNN') (Hopfield, 1982) layers. Conceptually, the lower layer, RNN_l, decodes word embeddings, and the upper layer, RNN_u, decodes tokens. A scalar 'stopness', s_t, gates the feed of the word embedding from the lower to the upper layer. s_t is generated by the upper layer. The lower hidden state is initialized from an input embedding, and the upper state is initialized as all zeros. At each time step:

Table 1: Example utterances from an instance of each grammar, for 4 different objects. Best viewed in color.

OBJECT	CONCAT	PERM	ROT	PROJ	SHUFDET	HOL
(0, 0, 0)	dadacbbba	aabdcdabb	ddccabcdd	dcbcbbaad	dadacbbba	adbcddadc
(0, 0, 1)	dadacbcca	aacdcdacb	ddccabdbb	bdbcabaad	ccadadacb	bcaadacba
(0, 1, 0)	dadcabbba	acbdadabb	ddcaabcdd	dbbcabcad	dadcabbba	bcaccaddb
(1, 0, 0)	ddbacbbba	aabdcbdbb	dcddbcdaa	acbcabaad	ddbacbbba	daaaacbdc

$$h_t^{(l)} = (1 - s_{t-1}) \cdot h_{t-1}^{(l)} + s_{t-1} \cdot \text{RNN}_u(\mathbf{0}, h_{t-1}^{(l)})$$
$$h_t^{(u)} = \text{RNN}_l\left(\mathbf{0}, (1 - s_{t-1}) \cdot h_{t-1}^{(u)} + s_{t-1} \cdot h_t^{(l)}\right)$$
$$s_t = \sigma(f_s(h_t^{(u)})) \qquad \hat{y}_t = o(h_t^{(u)})$$

where $o(\cdot)$ and $f_h(\cdot)$ are projection layers. HUSendA is an auto-regressive variant of HUSendZ, in which the input to RNN_l at each timestep is a projection of $\hat{y}_{t-1}$, instead of $\mathbf{0}$. Figure 5 depicts the HU-RNN Sender architecture graphically.

Note that we can choose any RNN for RNN_u and RNN_l. We use the suffix ':[rnn type]', where ':RNN' means a vanilla RNN, ':LSTM' is an LSTM, and ':dgsend' means using the Sender RNN from Dagan et al. (2020).

We also propose HURecv, which is a Receiver model, see Appendix C.5.

4 EXPERIMENTS

Code for experiments is provided at [1]

4.1 EXAMPLES OF GRAMMARS

Table 1 shows examples of each grammar, for 4 objects. For CONCAT, changing one attribute changes 3 adjacent utterance tokens. PERM rearranges columns of CONCAT utterance tokens. SHUFDET rearranges blocks of 3 utterance tokens, as a function of the last object attribute. We depict utterances for $n_{att} = 3$, and $c_{len} = 3 \cdot n_{att}$. In our experiments we use $n_{att} = 5$ and $c_{len} = 4 \cdot n_{att}$. Examples for this geometry can be found in Appendix E.

4.2 COMPOSITIONAL METRIC EVALUATION

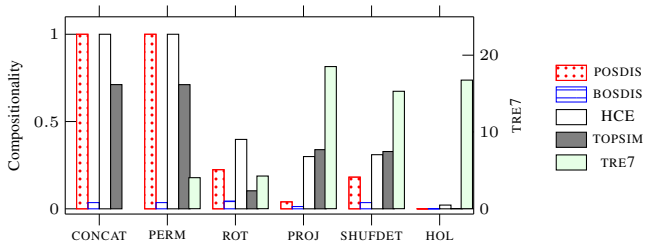

Figure 3: Values of compositionality metrics for each of the artificial grammars. Each bar is an average over 5 seeds. TRE7 uses right-hand y-axis. High TRE7 means low compositionality

Figure 3 shows the values of compositional metrics for samples from our artificial grammars, using $n_{att} = 5$, $n_{val} = 10$. The compositionality metrics show low compositionality for all the artificial grammars, except for CONCAT and PERM. Thus our transformations successfully hide the compositional structure from current compositional metrics.

[1] will be made available at publication, and is in the addendum during submission

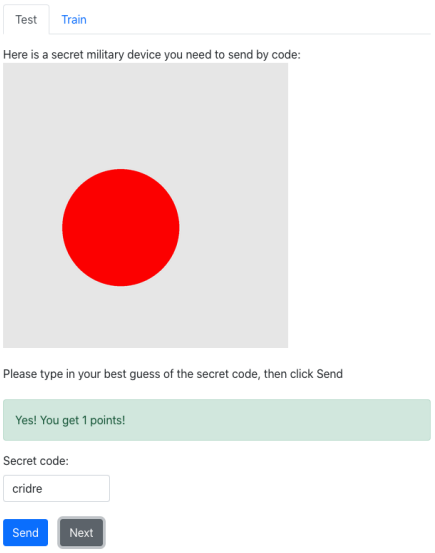

Figure 4: User interface for our 'Secret Spy Codes' game. This image depicts a PERM grammar, where the original utterance was 'redcir'.

4.3 HUMAN EVALUATION

We constructed an MTurk (Crowston, 2012) task, 'Secret Spy Codes', in order to evaluate human performance on ICY grammars. Figure 4 shows the user interface. Human subjects were tasked with writing out the code that represents depicted geometric objects. They needed substantial effort in order to learn just a few new words. Thus, we used objects with only two attributes: shape and color; and we experimented with using abbreviated English words, which were easier to learn.

4.3.1 DATASET

SYNTH uses artificially generated random words for each attribute value. We sample 2-letter words from a vocabulary size of 4. Each utterance therefore has 4 letters: 2 for shape, and 2 for color. Since humans found these words challenging to learn, so we used just three possible values for each attribute, giving 9 combinations in total.

ENG uses 3-letter English abbreviations for attribute values, e.g. 'tri' for 'triangle', and 'grn' for 'green'. The words describing each attribute value in ENG are relatively easy for a human to learn. Therefore, we used 5 attribute values for each attribute, giving 25 possible combinations.

We held out three color-shape combinations, that were not made available during training. For example, subjects might have access to a red circle and a blue triangle, but not a red triangle. Subjects who could perceive the compositional structure of a grammar should be able to get these holdout instances correct.

4.3.2 EVALUATION

Dataset	CONCAT	PERM	PROJ	ROT	SHUFDET
SYNTH	$\mathbf{0.2 \pm 0.1}$	0.04 ± 0.05	0.02 ± 0.04	0.04 ± 0.07	0.06 ± 0.06
ENG	$\mathbf{0.6 \pm 0.2}$	0.2 ± 0.2	0.04 ± 0.08	0.02 ± 0.03	$\mathbf{0.7 \pm 0.2}$

Table 2: Human evaluation results. Values are $\text{acc}_{holdout}$

We measured subjects' accuracy on the 3 held out examples. The results are shown in Table 2. For SYNTH, $\text{acc}_{holdout}$ was poor for all grammars: humans were unable to spot compositional form using unfamiliar words. In ENG, $\text{acc}_{holdout}$ was high for both CONCAT and SHUFDET grammars,

Table 3: Compositional inductive bias for some standard Sender models, for $n_{att} = 5$, $n_{val} = 10$. Results are each averaged over 10 seeds. Results are the ratio of the convergence time for the grammar relative to CONCAT, therefore have no units. CI95 is in Table 12.

Model	Params	PERM	PROJ	ROT	SHUFDET	HOL
1-layer MLP (Rumelhart et al., 1986)	5100	1.12	1.9	> 20	> 20	> 20
2-layer MLP	19428	1	2.1	> 20	7	> 20
1-layer LSTM (Hochreiter & Schmidhuber, 1997)	139909	1	2.2	7	1.6	> 20
2-layer LSTM	272005	0.9	1.9	4.6	1.36	> 20
1-layer Transformer decoder (Vaswani et al., 2017)	272389	1.02	2.5	10	2.1	> 20
2-layer Transformer decoder	536965	1.08	2.2	10.4	1.98	> 20
Hashtable	O(N)	1	0.98	0.98	1.02	1.06

Table 4: Effect of number of parameter on compositional inductive bias. Results are each averaged over 5 seeds. CI95 is in Table 13

Model	Emb size	Params	PERM	PROJ	ROT	SHUFDET	HOL
1-layer LSTM	128	139909	1	2.2	7	1.6	> 20
1-layer LSTM	1280	13195525	0.94	1.6	2.7	1.4	> 20
2-layer MLP	128	19428	1	2.1	> 20	7	> 20
2-layer MLP	1280	193380	1.02	1.86	> 20	10	> 20

as expected, and low for all other grammars. This shows that the composition functions in PERM, PROJ and ROT were not clearly apparent to human subjects, even though, as we shall see next, neural models can acquire these grammars easily.

4.4 NEURAL MODEL EVALUATION

We use the ICY benchmark to evaluate standard neural models for specific aspects of their compositional inductive bias. We focus on Sender models in our presentation. Results for Receiver models are in Appendix G. We train each model supervised on a specific artificial grammar from ICY, using cross-entropy loss.

We count the number of training steps, $N_{acquire}$, required to train each grammar to a training accuracy of acc_{tgt}, where accuracy is token-level accuracy. For each grammar, $\mathcal{G}$, we report the ratio $b^{(\mathcal{G})} = N^{(\mathcal{G})}_{acquire}/N^{(\mathcal{G}_{\text{CONCAT}})}_{acquire}$. We used $n_{att} = 5$ and $n_{val} = 10$, $c_{len} = 20$, $V = 4$, and $\text{acc}_{tgt} = 0.8$. We halt training if $b^{(\mathcal{G})}$ reaches 20.

Table 3 shows the results. Detailed architectural descriptions of the 'Model' column are provided in Appendix B. The remaining columns, except for 'Params', show the acquisition time, b, for each grammar, relative to CONCAT. We have highlighted in red the scenarios that failed to reach convergence; and in green the scenarios where b was less than 1/3 that of HOL, which shows that language acquisition was relatively fast.

We can see that for many models, our transformations do not much affect the acquisition speed by neural networks. Therefore, in an emergent communication scenario, neural models can generate languages which appear non-compositional both to our current metrics, and to human evaluation. Such languages will therefore be deemed 'non-compositional' by all current evaluation methods, except for generalization. This might explain the empirically observed lack of correlation between measured language compositionality, and generalization, in emergent communication experiments.

4.5 RESULTS ARE INDEPENDENT OF NUMBER OF PARAMETERS

An obvious concern with Table 3 is that the number of parameters varies between models, so we vary the parameters, by changing the hidden size. Table 4 shows the results. We can see that the relative acquisition speed, relative to CONCAT, is not changed much by a 10-fold increase in parameters, relative to the differences between the architectures. This is encouraging: we are not simply viewing an artifact of model size.

Table 5: Zero-RNN improves bias against PERM. Results are mean over 5 runs. CI95 Table 14.

Model	Parameters	PERM	PROJ	ROT	SHUFDET	HOL
RNN	40837	0.76	2.8	18	1.74	>20
RNN-Z	40069	0.8	2.9	19	2	>20
GRU (Cho et al., 2014)	106885	1	2.4	6	1.88	>20
GRU-Z	106117	1.16	2.3	4.7	1.86	>20
LSTM	139909	1	2.2	7	1.6	>20
LSTM-Z	139141	1.18	2.3	7	1.82	>20

Table 6: Selected results for low SHUFDET bias, mean over 10 runs. Full results Table 15.

Model	Params	PERM	PROJ	ROT	SHUFDET	HOL
LSTM	139909	1.08	2.4	6.3	1.8	> 20
dgsend	106117	0.92	2.3	5.5	1.52	> 20
HUSendZ:RNN	40710	0.93	**2.5**	**15**	1.68	> 20
HUSendZ:dgsend	138758	1.03	1.67	4.5	**1.27**	> 20

4.6 RNNZERO INCREASES BIAS AGAINST PERM

Table 5 shows the effect of not feeding the output of an RNN decoder as the input at each step. Surprisingly this increases bias against PERM. That is, actually increases a prior over adjacency.

4.7 HUSENDZ:DGSEND HAS LOW BIAS AGAINST SHUFDET

We searched for neural models with reduced bias against SHUFDET, including using RNN-Z, dgsend (Dagan et al., 2020), and HU-RNN. Table 6 shows a sub-set of the results. More results are in Appendix H. 'dgsend' acquired SHUFDET faster than LSTM. HUSend using a vanilla RNN as RNN_l and RNN_u acquired SHUFDET faster than LSTM. Combining HUSendZ with dgsend acquired SHUFDET fastest.

4.8 END-TO-END TRAINING

We experimented with measuring the compositional inductive bias of a Sender and Receiver model placed end to end, see Appendix I

5 CONCLUSION

We have shown that it is possible to construct transformations that, when applied to concatenation grammars, result in grammars that machines can learn easily but which humans find challenging to learn. This could explain the disconnect highlighted in recent papers between neural network ability to generalize, in an emergent communication context, and the compositionality of the resulting languages, as measured by recent metrics of compositionality. We propose to use the families of transformations as a benchmark, ICY, for measuring aspects of the compositional inductive bias of neural networks, and searching for models with similar biases to humans. We use our benchmark to propose one such neural model, HU-RNN, which shows a compositional inductive bias towards relocatable atomic word-like groups of tokens.

6 REPRODUCIBILITY

Full code is provided in the addendum, along with instructions in the README.md. Full code will be published to github following acceptance. Each experiment was run multiple times (usually 5 or 10), using different seeds, and the mean reported. CI95 ranges are available in Appendix F.

7 ETHICS

This work does involve human subjects, who needed to learn to use artificially generated codes to label abstract geometric objects. The annotation device was created as a game, that many people found fun to play. We received many feedbacks stating 'good', 'very interesting task'. None of the language or figures being trained on contain any obvious characteristics which could be deemed racist, sexist, or having any other obvious human-centric harmful biases, as far as we can tell.

This work contains no obviously harmful insights, methodologies or applications. There are no obvious conflicts of interest or sponsorship to note. There are no obvious discrimination/bias/fairness concerns to report. There are no obvious issues with privacy, security, or legal compliance. All data provided was artificially generated, and does not present privacy or other issues. We have done our due diligence to ensure the integrity and reproducibility of our research.

Although emergent communication investigates the communications between neural models, who learn to generate new languages, as part of collaborative tasks, we do not believe that such models are 'alive', or 'conscious', though we admit that we do not have any way to determine this in any objective way. The number of neurons of the models concerned was orders of magnitude less than that of the human brain. The models were not exposed to sufficiently varied or complex data that we feel that they could have learned advanced sentience or perception, although again we admit that we are not aware of an objective 'threshold' or similar that we could compare with.

REFERENCES

Jacob Andreas. Measuring compositionality in representation learning. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HJz05o0qK7.

Robert Berwick, Gabriel Beckers, Kazuo Okanoya, and Johan Bolhuis. A bird's eye view of human language evolution. *Frontiers in evolutionary neuroscience*, 4:5, 2012.

Henry Brighton and Simon Kirby. Understanding linguistic evolution by visualizing the emergence of topographic mappings. *Artificial life*, 12(2):229–242, 2006.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Rahma Chaabouni, Eugene Kharitonov, Diane Bouchacourt, Emmanuel Dupoux, and Marco Baroni. Compositionality and generalization in emergent languages. *arXiv preprint arXiv:2004.09124*, 2020.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans (eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pp. 1724–1734. ACL, 2014. doi: 10.3115/v1/d14-1179. URL https://doi.org/10.3115/v1/d14-1179.

Kevin Crowston. Amazon mechanical turk: A research tool for organizations and information systems scholars. In Anol Bhattacherjee and Brian Fitzgerald (eds.), *Shaping the Future of ICT Research. Methods and Approaches*, pp. 210–221, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-35142-6.

Gautier Dagan, Dieuwke Hupkes, and Elia Bruni. Co-evolution of language and agents in referential games. *arXiv preprint arXiv:2001.03361*, 2020.

Jakob Foerster, Ioannis Alexandros Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 29, pp. 2137–2145. Curran Associates, Inc., 2016.

Thomas L Griffiths and Michael L Kalish. Language evolution by iterated learning with bayesian agents. *Cognitive science*, 31(3):441–480, 2007.

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

John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. Compositionality decomposed: How do neural networks generalise? *Journal of Artificial Intelligence Research*, 67:757–795, 04 2020. doi: 10.1613/jair.1.11674.

Eugene Kharitonov and Marco Baroni. Emergent language generalization and acquisition speed are not tied to compositionality. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pp. 11–15, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.blackboxnlp-1.2. URL https://aclanthology.org/2020.blackboxnlp-1.2.

Simon Kirby, Hannah Cornish, and Kenny Smith. Cumulative cultural evolution in the laboratory: An experimental approach to the origins of structure in human language. *Proceedings of the National Academy of Sciences*, 105(31):10681–10686, 2008.

Andrei N Kolmogorov. On tables of random numbers. *Sankhyā: The Indian Journal of Statistics, Series A*, pp. 369–376, 1963.

Satwik Kottur, José Moura, Stefan Lee, and Dhruv Batra. Natural language does not emerge 'naturally' in multi-agent dialog. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 2962–2967, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1321. URL https://www.aclweb.org/anthology/D17-1321.

Angeliki Lazaridou, Karl Moritz Hermann, Karl Tuyls, and Stephen Clark. Emergence of linguistic communication from referential games with symbolic and pixel input. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=HJGv1Z-AW.

Tao Lei, Yu Zhang, Sida I. Wang, Hui Dai, and Yoav Artzi. Simple recurrent units for highly parallelizable recurrence. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

David Lewis. *Convention: A philosophical study*. John Wiley & Sons, 2008.

Fushan Li and Michael Bowling. Ease-of-teaching and language structure from emergent communication. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 15825–15835, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/b0cf188d74589db9b23d5d277238a929-Abstract.html.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc,

E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Steven Pinker and Paul Bloom. Natural language and natural selection. *Behavioral and brain sciences*, 13(4):707–727, 1990.

Cinjon Resnick, Abhinav Gupta, Jakob Foerster, Andrew M. Dai, and Kyunghyun Cho. Capacity, bandwidth, and compositionality in emergent language learning. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '20, pp. 1125–1133, Richland, SC, 2020. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450375184.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, undefinedukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Jennifer C. White and Ryan Cotterell. Examining the inductive bias of neural language models with artificial languages. 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)*, pp. 454–463, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.38. URL https://aclanthology.org/2021.acl-long.38.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pp. 11328–11339. PMLR, 2020.

A HYPERPARAMETERS

A.1 GENERAL HYPER-PARAMETERS

Table 7: General hyper-parameters, for sender and receiver network experiments

Setting	Value
Embedding size, d_{emb}	128
Vocab size, V	4
Utterance length, c_{len}	4 * n_{att}
Dropout	0
Gradient clipping	5.0
Optimizer	Adam
Batch size	128

General hyper-parameters are shown in Table 7.

B SENDER MODEL ARCHITECTURES

We use a separate embedding matrix for each attribute, where the number of embeddings is equal to n_{val}. Given an object with n_{att} attributes, we embed each of the attributes, then take the sum, to form a vector $e \in \mathbb{R}^{d_{emb}}$

B.1 1-LAYER MLP

Instead of embedding into $e \in \mathbb{R}^{d_{emb}}$, we embed into $\mathbb{R}^{c_{len} \cdot V}$, then we reshape into $\mathbb{R}^{c_{len} \times V}$.

B.2 2-LAYER MLP

We form $W^T \tanh(\text{drop}(e))$, where W is a learnable matrix $\in \mathbb{R}^{d_{emb} \times (c_{len} \cdot V)}$. Then we reshape to be $\in \mathbb{R}^{c_{len} \times V}$

B.3 1-LAYER LSTM

We apply dropout to the embeddings e, then we use as the initial hidden state for the LSTM. At each timestep, we project the output token from the previous timestep (initially zero), and pass as the input token. We project the output at each timestep, to be in $\mathbb{R}^V$, and form the softmax, to obtain a probability distribution over tokens.

B.4 2-LAYER LSTM

2-layer version of the 1-layer LSTM above, where the output of the first layer at each timestep is fed into the input of the second layer. Each layer has its own hidden state and cell state. We project the output from the second layer at each timestep, to be in $\mathbb{R}^V$, and form the softmax, to obtain a probability distribution over tokens.

B.5 1- OR 2-LAYER TRANSFORMER DECODER

TransDecSoft is a transformer decoder, as defined in Vaswani et al. (2017). Each softmaxed output token is passed in as input to the following timestep.

B.6 HASHTABLE

Hashtable is a standard hashtable. We trained and scored using a similar approach to neural nets:

- A minibatch of training examples was presented to the hashtable.

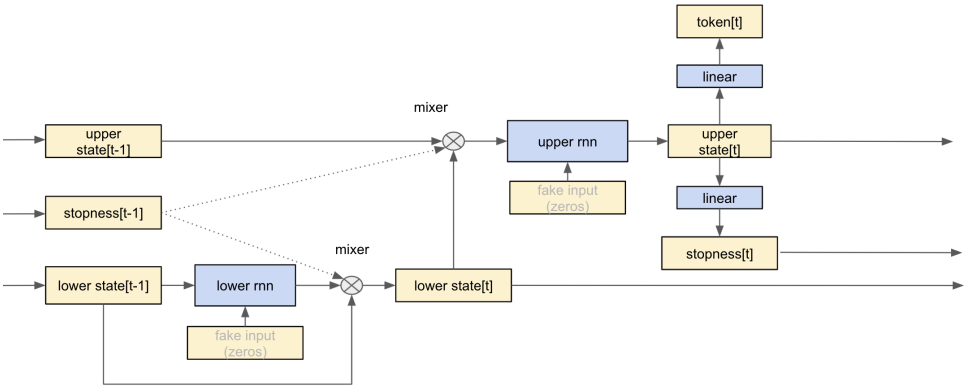

Figure 5: HU-RNN Sender Architecture

- The hashtable made a prediction. For previously unseen inputs, the hashtable predicted all 0s.
- The training accuracy was calculated using these predictions.
- The examples from this minibatch were added to the hashtable.

B.7 HIERARCHICAL-UNIT RNN SENDER, HU-SENDER

Figure 5 depicts the HU-RNN Sender architecture graphically.

C RECEIVER MODEL ARCHITECTURES

Given an input utterance of length c_{len}, vocab size V, in all cases, we first embed the tokens, to form a tensor $e \in \mathbb{R}^{c_{len} \times d_{emb}}$.

C.1 CNN

4 convolutional blocks, where each block consists of:

- embed, as above
- 1d convolution (kernel size 3, padding 1, stride 1)
- max pooling (kernel size 2, padding 0, stride 2)
- ReLU activation

We only experiment with using a CNN as a receiver network.

C.2 FC2L

- embed, as above, to form e
- form vec$(\tanh(\text{drop}(e)))$
- project, using learnable matrix W
- reshape to be $\in R^{n_{att} \times n_{val}}$

C.3 RNNXL:RNNTYPE

Synonym for rnntype-xL, e.g. RNN2L:LSTM is equivalent to LSTM-1L. We first embed to form e then pass the embedding for each timestep $t \in \{1, \ldots, c_{len}\}$ into the RNN at each timestep. We take the final hidden state, apply dropout, and project using learnable matrix W to be in $\mathbb{R}^{n_{att} \times n_{val}}$.

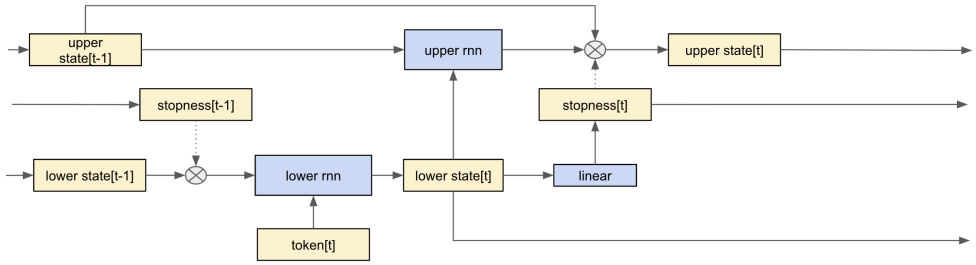

Figure 6: HU-RNN Receiver Architecture

C.4 SRU

See Lei et al. (2018). An SRU is derived from an LSTM but with no connection between the hidden states. Connections between cell states remain. An SRU is much faster than an LSTM to train on a GPU. 'RNN2L:SRU' is a synonym for a 2-layer SRU, i.e. SRU-2L. We treat SRU as any other RNN, see above.

C.5 HIERARCHICAL-UNIT RNN RECEIVER, HU-RECEIVER

HU-Sender was described in the main body.

HU-Receiver is a fully-differentiable hierarchical receiver. Incoming tokens, x_t are fed in to a lower RNN, RNN_l, one token per time step. The output state $h_t^{(l)}$ of RNN_l at each time step is used to generate a scalar 'stopness', s_t, which conceptually represents the end of a word-like group of tokens. The upper state $h_t^{(u)}$ conceptually copies $h_{t-1}^{(u)}$ when s_t is near 0, or takes a step using an upper RNN, RNN_u, when s_t is near 1. The formulae are thus:

$$
\begin{aligned}
h_t^{(l)} &= \text{RNN}_l(i(x_t), h_{t-1}^{(l)}) \\
s_t &= \sigma(f_s(h_t^{(l)})) \\
\tilde{h}_t^{(u)} &= \text{RNN}_u(h_t^{(l)}, h_{t-1}^{(u)}) \\
h_t^{(u)} &= (1 - s_t) \cdot h_t^{(u)} + s_t \cdot \tilde{h}_t^{(u)}
\end{aligned}
$$

where $i(\cdot)$ and $f_s(\cdot)$ are projection layers, and $\sigma(\cdot)$ is the sigmoid function.

Figure 6 depicts the HU-RNN Receiver model graphically.

C.6 HIER:RNNTYPE

Synonym for the HU-Receiver model, described above. 'rnntype' describes the RNN model used for each of RNN_l and RNN_u.

D HUMAN COMPOSITIONAL ENTROPY METRIC

D.1 EXPOSITION

We relax the minimization in the Residual Entropy metric (Resnick et al., 2020). Resnick defines residual entropy as:

$$
re(\mathcal{M}, \mathcal{O}) = \min_{p \in \mathcal{P}} \frac{1}{n_{att}} \sum_{i=1}^{n_{att}} \frac{\mathcal{H}_{\mathcal{M}}(o^{(i)}|z[p_i])}{\mathcal{H}_{\mathcal{O}}(o^{(i)})}
$$

Table 8: Residual entropy metrics we compare

Metric	Description
resent_ours	Our implementation of residual entropy, including exhaustive search over partitions, and with optional normalization
resent_resnick	Implementation of residual entropy in code for Resnick et al. (2020), which uses a greedy approach, but requires $V = 2$. It does not normalize. We generalized this to work for arbitrary n_{att}; and modified it to return base-2 entropy
resent_relax = 1 − HCE	Our relaxed version of residual entropy, works for arbitrary n_{att} and V, optional normalization

where $\mathcal{M}$ is the space of messages, i is an index over attributes, $o^{(i)}$ is the i'th attribute, $\mathcal{P}$ is the set of all possible partitions over the messages, p is one such partition, p_i is one set in the partition p, $z[p_i]$ is the sub-sequence of each message indexed by set p_i, and $\mathcal{H}$ is the entropy. Thus, residual entropy finds a partition over messages, into n_{att} sets, which associates each set in the partition with a specific attribute $\in \{1, \ldots, n_{att}\}$, and minimizes the conditional entropy between each attribute in the dataset, and the corresponding message sub-sequences.

We can see that residual entropy assumes a composition that comprises a permutation over concatenation. This is thus a measure of human compositionality. It does not attempt to measure other members of the class of generalizable composition functions.

The minimization over $p \in \mathcal{P}$ is problematic because it involves a minimization over a combinatorial number of partitions. We seek to relax this, by using a greedy approach.

Similar to Chaabouni et al. (2020) we form $\mathcal{I}(m^{(j)}; o^{(i)})$, the mutual information between the j'th symbol of each message, $m^{(j)}$, and the i'th attribute of each object, $o^{(i)}$, over the entire dataset:

$$\mathcal{I}(m^{(j)}; o^{(i)}) = \sum_{n=1}^{N} p(m_n^{(j)}, o_n^{(i)}) \log \frac{p(m_n^{(j)}, o_n^{(i)})}{p(m_n^{(j)}) p(o_n^{(i)})}$$

For each $m^{(j)}$, we calculate $o^{(j^*)} = \arg\max_{o^{(i)}} \mathcal{I}(m^{(j)}; o^{(i)})$. That is, $o^{(j^*)}$ is the attribute that has the highest mutual information with $m^{(j)}$. This defines a partition over messages. For each attribute $o^{(i)}$, the associated message sub-sequence is $p_i = \{m^{(j)} | o^{(j^*)} = o^{(i)}, \forall o^{(i)}\}$.

Thus, given language $\mathcal{G}$, we calculate HCE as:

$$\text{HCE}(\mathcal{G}) = 1 - \frac{1}{n_{att}} \sum_{i=1}^{n_{att}} \frac{H(o^{(i)} \mid p_i)}{H(o^{(i)})} \tag{1}$$

where we subtract from 1, so that an HCE of 1 means compositional, and 0 means non-compositional, in alignment with other compositionality metrics, such as topsim, bosdis, posdis. To avoid confusion, we give the resulting metric a new name 'Human Compositional Entropy', abbreviated as 'HCE'.

HCE has similar speed advantages to posdis and bosdis, but assumes only $c_{len} \geq n_{att}$. posdis and bosdis provide alternative relaxations of residual entropy, but they both require that $c_{len} = n_{att}$. HCE lies in $[0, 1]$, in alignment with topsim, bosdis, and posdis. We present empirical comparisons between resent and HCE next.

D.2 Empirical comparison of residual entropy and HCE

We compare the metrics shown in Table 8.

resent_ours is as far as we know a correct implementation of the residual entropy algorithm in Resnick et al. (2020). The result can optionally be normalized. Unfortunately, exhaustively searching over all possible partitions of the messages takes combinatorial time, and becomes unworkably

Table 9: Comparison of metric results for resent_resnick and resent_relax, using $V = 2$, $n_{att} = 2$, $n_{val} = 5$, $c_{len} = 10$. No normalization. resent_relax$= 1 -$ HCE

Grammar	resent_ours	resent_resnick	resent_relax
COMP	0.0000	0.0000	0.0000
PERM	0.0000	0.0000	0.0000
PROJ	0.0000	0.8468	0.8333
SHUFDET	0.0000	0.4796	0.6452
SHUF	0.0000	0.5600	0.5389
ROT	0.0000	0.5510	0.5510
HOL	0.0000	0.5616	0.5728

Table 10: Comparison of metric results for resent_ours and resent_relax, using $V = 4$, $n_{att} = 3$, $n_{val} = 4$, $c_{len} = 6$. Normalization enabled.

Grammar	resent_ours	resent_relax
COMP	0.0000	0.0000
PERM	0.0000	0.0000
PROJ	0.4772	0.5343
SHUFDET	0.2337	0.4025
SHUF	0.4340	0.4973
ROT	0.0814	0.3867
HOL	0.4954	0.6183

slow for high n_{att} and high c_{len}. resent_resnick is our fork of the code in [2], which we have modified to work with arbitrary n_{att}, and to use base-2 entropy. It uses a greedy approach, but requires $V = 2$, as far as we know. It does not normalize the result. resent_relax$= 1 - HCE$ is our relaxed version of residual entropy, but without subtracting from 1. The result can optionally be normalized.

We first compare all three metrics. This requires using $V = 2$, to satisfy resent_resnick, low n_{att}, to keep the calculation time for resent_ours reasonable, and high c_{len} to make it possible to construct a COMP grammar with a small V. We disable normalization, since resent_resnick does not implement it. Table 9 shows the results, which are each averaged over 5 seeds. We can see that resent_ours consistently scores 0, over all languages. This is probably because the utterance length is so long that there are many possible partitions, of which at least one gives zero entropy. resent_resnick and resent_relax give similar results, except for SHUFDET where resent_resnick gives a lower score than resent_relax.

Then, we increase the vocabulary size V. This precludes measuring resent_resnick, which requires $V = 2$, but allows for a shorter c_{len} and higher n_{att}. We enable normalization, since both metrics support it. Table 10 depicts the results. In these conditions, resent_ours is non-zero wherever resent_relax is non-zero. resent_relax returns results which are higher than resent_ours, but do correlate somewhat. The values of resent_relax for each grammar appear plausible, e.g. that for HOL is higher than for other grammars.

E EXAMPLE UTTERANCES

Table 11 depicts example utterances for $n_{att} = 5$ and $c_{len} = 4 \cdot n_{att}$.

F CI95 VALUES FOR KEY TABLES

Table 12 shows the sender 10^5 results, including CI95 ranges, i.e. for Table 3. Note that for any training runs that were truncated at a ratio of 20, the variance will appear to be 0, as long as all runs were truncated at a ratio of 20.

[2] https://github.com/backpropper/cbc-emecom/blob/6d01f0cdda4a8f742232b537da4f2633613f44a9/utils.py#L164-L204

Table 11: Example utterances for 4 objects, using $n_{att} = 5$ and $c_{len} = 4 \cdot n_{att}$.

	Objects (0, 0, 0, 0, 0)	(0, 0, 0, 0, 1)	(0, 0, 0, 1, 0)	(0, 0, 1, 0, 0)
concat	cdbacadcbcacbddacadb	cdbacadcbcacbddacadb	cdbacadcbcacbddacadb	cdbacadcbcacbddacadb
perm	dabdccbdcacabddacabc	dabdccbdcacabddacabc	dabdccbdcacabddacabc	dabdccbdcacabddacabc
rot	cbccaadbcaacdcbbddcd	cbccaadbcaacdcbbddcd	cbccaadbcaacdcbbddcd	cbccaadbcaacdcbbddcd
proj	bbdaacdaadbbabbcbbcb	bbdaacdaadbbabbcbbcb	bbdaacdaadbbabbcbbcb	bbdaacdaadbbabbcbbcb
shufdet	bcaccadcbddacdbacadb	bcaccadcbddacdbacadb	bcaccadcbddacdbacadb	bcaccadcbddacdbacadb
hol	bdabaabbacacdbaabdcd	bdabaabbacacdbaabdcd	bdabaabbacacdbaabdcd	bdabaabbacacdbaabdcd

Table 13 shows the CI95 ranges for the additional results shown in Table 4 (the additional rows in Table 4 were copied from Table 3).

Table 14 shows the CI95 ranges for the results shown in Table 5.

Table 15 shows the full results for the search for low SHUFDET bias, including CI95 ranges.

arch	params	Permute ratio	RandomProj ratio	Cumrot ratio	ShuffleWordsDet ratio	Holistic ratio
FC1L	5100	1.12+/-0.09	1.9+/-0.2	20.000+/-0.000	11+/-6	20.000+/-0.000
FC2L	19428	1.000+/-0.000	2.1+/-0.2	20.000+/-0.000	7+/-3	20.000+/-0.000
RNNAutoReg:LSTM	139909	1.00+/-0.10	2.2+/-0.2	7+/-2	1.60+/-0.08	20.08+/-0.04
RNNAutoReg2L:LSTM	272005	0.9+/-0.2	1.9+/-0.3	5+/-1	1.4+/-0.2	20.06+/-0.04
TransDecSoft	272389	1.0+/-0.2	2.5+/-0.6	10+/-3	2.1+/-0.4	20.4+/-0.1
TransDecSoft2L	536965	1.08+/-0.07	2.2+/-0.5	10.4+/-1.0	2.0+/-0.1	20.30+/-0.06
Hashtable	0	1.0+/-0.1	0.98+/-0.04	0.98+/-0.04	1.02+/-0.09	1.1+/-0.1

Table 12: CI95 ranges for Sender model results.

arch	params	Permute ratio	RandomProj ratio	Cumrot ratio	ShuffleWordsDet ratio	Holistic ratio
RNNAutoReg:LSTM	13195525	0.9+/-0.2	1.6+/-0.3	2.7+/-0.6	1.4+/-0.2	20.000+/-0.000
FC2L	193380	1.02+/-0.07	1.9+/-0.1	20.000+/-0.000	10+/-5	20.000+/-0.000

Table 13: CI95 ranges for effect of number of parameters.

arch	params	Permute ratio	RandomProj ratio	Cumrot ratio	ShuffleWordsDet ratio	Holistic ratio
RNNAutoReg:RNN	40837	0.76+/-0.09	2.80+/-0.10	18+/-3	1.7+/-0.1	20.06+/-0.04
RNNZero:RNN	40069	0.80+/-0.10	2.9+/-0.7	19+/-2	2.0+/-0.2	20.000+/-0.000
RNNAutoReg:GRU	106885	1.0+/-0.1	2.4+/-0.3	6+/-1	1.9+/-0.2	20.12+/-0.04
RNNZero:GRU	106117	1.2+/-0.2	2.3+/-0.3	4.7+/-0.5	1.9+/-0.1	20.04+/-0.04
RNNAutoReg:LSTM	139909	1.00+/-0.10	2.2+/-0.2	7+/-2	1.60+/-0.08	20.08+/-0.04
RNNZero:LSTM	139141	1.18+/-0.07	2.3+/-0.3	7+/-2	1.8+/-0.1	20.06+/-0.04

Table 14: CI95 ranges for performance of ZeroRNN.

arch	params	Permute ratio	RandomProj ratio	Cumrot ratio	ShuffleWordsDet ratio	Holistic ratio
RNNAutoReg:LSTM	139909	1.1+/-0.1	2.4+/-0.3	6+/-1	1.8+/-0.1	20.04+/-0.03
RNNZero:LSTM	139141	1.1+/-0.1	2.3+/-0.2	5.4+/-1.0	1.8+/-0.1	20.04+/-0.03
HierAutoReg:RNN	64262	0.92+/-0.05	2.4+/-0.3	10+/-3	1.9+/-0.2	20.10+/-0.06
HierZero:RNN	40710	0.9+/-0.1	2.5+/-0.4	15+/-2	1.7+/-0.2	20.09+/-0.04
RNNAutoReg:dgsend	106117	0.92+/-0.10	2.3+/-0.3	5.5+/-0.8	1.5+/-0.1	20.13+/-0.04
RNNZero:dgsend	105349	0.92+/-0.08	2.1+/-0.2	4.6+/-0.4	1.5+/-0.1	20.11+/-0.04
HierAutoReg:dgsend	178566	1.09+/-0.07	1.9+/-0.2	6+/-1	1.4+/-0.1	20.11+/-0.04
HierZero:dgsend	138758	1.03+/-0.08	1.7+/-0.1	4.5+/-0.8	1.3+/-0.1	20.11+/-0.04

Table 15: Full table for SHUFDET, including CI95 ranges.

G ADDITIONAL RESULTS

Table 16: Relative compositional inductive biases for sender models, for $n_{att} = 5$, $n_{val} = 10$. $\uparrow$ and $\downarrow$ denotes columns we want to maximize or minimize respectively.

Repr	Model	CONCAT	PROJ $\downarrow$	PAIRSUM $\downarrow$	PERM $\downarrow$	ROT $\downarrow$	SHUFDET $\uparrow$	HOL $\downarrow$
SOFT	FC1L	1.000	0.88	0.538	1.000	**0.49**	0.77	0.255
	FC2L	1.000	0.924	0.54	1.000	**0.49**	0.77	0.253
	HierZero:RNN	0.995	0.846	0.629	0.995	0.68	0.95	**0.242**
	HierAutoReg:RNN	0.995	0.87	0.68	0.987	0.74	**0.985**	0.254
	RNNZero2L:SRU	0.993	**0.790**	0.76	**0.971**	0.72	0.931	0.249
	TransDecSoft	0.994	0.812	**0.53**	0.984	0.58	0.82	0.249
	TransDecSoft2L	0.995	0.818	0.61	0.990	0.548	0.85	0.246
GUMB	FC1L	0.997	0.846	0.53	1.000	**0.49**	0.750	0.249
	FC2L	0.998	0.84	**0.49**	1.000	**0.49**	0.74	0.256
	HierZero:RNN	0.991	0.85	0.64	0.989	0.72	**0.96**	0.249
	HierAutoReg:RNN	0.994	0.858	0.63	0.993	0.68	0.94	0.256
	RNNZero2L:SRU	0.992	**0.76**	0.722	**0.95**	0.691	0.89	0.255
	TransDecSoft	0.993	0.78	0.53	0.981	0.53	0.81	0.251
	TransDecSoft2L	0.992	0.784	0.52	0.979	0.50	0.77	**0.243**
DISCR	FC1L	0.961	0.739	0.423	0.984	0.46	0.67	0.252
	FC2L	0.969	0.75	0.42	0.981	0.459	0.68	0.246
	HierZero:RNN	0.959	0.731	0.52	0.93	0.61	0.70	0.249
	HierAutoReg:RNN	0.956	**0.65**	0.53	0.94	0.607	**0.74**	0.248
	RNNZero2L:SRU	0.955	0.67	0.629	**0.87**	0.65	0.715	**0.241**
	TransDecSoft	0.956	0.68	0.364	0.93	0.49	0.63	0.255
	TransDecSoft2L	0.951	0.67	**0.304**	0.91	**0.42**	0.45	0.245

Table 17: Relative compositional inductive biases for receiver models, for $n_{att} = 5$, $n_{val} = 10$. $\uparrow$ and $\downarrow$ denotes columns we want to maximize or minimize respectively.

Repr	Model	CONCAT	PROJ $\downarrow$	PAIRSUM $\downarrow$	PERM $\downarrow$	ROT $\downarrow$	SHUF $\uparrow$	SHUFDET $\uparrow$	HOL $\downarrow$
SOFT	CNN	0.997	0.57	0.59	0.965	0.92	0.63	0.82	0.106
	FC2L	1.000	0.77	0.50	1.000	**0.44**	0.63	0.84	0.101
	Hier:GRU	0.996	0.57	0.51	0.95	0.94	**0.972**	0.970	0.111
	Hier:dgrecv	0.993	**0.52**	0.43	0.972	0.96	0.89	0.91	0.097
	RNN1L:LSTM	0.994	0.529	0.42	0.93	0.83	0.721	0.88	**0.091**
	RNN1L:dgrecv	0.993	0.532	**0.38**	0.963	0.90	0.70	0.91	0.096
	RNN2L:GRU	0.996	0.568	0.57	0.93	0.94	0.90	**0.973**	0.112
	RNN2L:SRU	0.994	0.55	0.47	**0.91**	0.89	0.90	0.962	0.099
GUMB	CNN	0.994	0.53	0.49	0.93	0.79	0.58	0.85	0.107
	FC2L	1.000	0.657	0.37	1.000	**0.38**	0.485	0.75	0.107
	Hier:GRU	0.994	0.54	0.46	0.96	0.943	**1.000**	0.999	0.099
	Hier:dgrecv	0.998	0.54	0.50	0.963	0.982	**1.000**	**1.000**	0.096
	RNN1L:LSTM	0.997	0.511	0.30	0.977	0.84	0.85	0.937	**0.094**
	RNN1L:dgrecv	0.996	0.62	0.61	1.000	0.95	0.88	0.93	0.103
	RNN2L:GRU	0.995	0.54	0.42	0.95	0.95	0.94	0.981	0.10
	RNN2L:SRU	0.995	**0.45**	**0.28**	**0.87**	0.80	0.947	0.967	0.100
DISCR	CNN	0.8	**0.28**	0.32	**0.7**	0.59	0.5	0.70	0.098
	FC2L	0.93	0.67	0.44	0.96	**0.47**	0.48	0.69	0.103
	Hier:GRU	0.973	0.49	0.43	0.981	0.8	**1.000**	**0.98**	0.097
	Hier:dgrecv	0.96	0.508	0.47	0.976	0.6	0.7	0.93	0.096
	RNN1L:LSTM	0.989	0.44	0.25	0.90	0.568	0.77	0.90	0.100
	RNN1L:dgrecv	0.93	0.59	0.5	0.991	0.87	0.989	0.95	0.120
	RNN2L:GRU	0.986	0.46	0.37	0.93	0.80	0.92	**0.98**	0.100
	RNN2L:SRU	0.961	0.36	**0.21**	0.77	0.60	0.82	0.85	**0.094**

Tables 16 and 17 show more results for both sender and receiver models, trained supervised in isolation. We are using the evaluation method here of measuring the number of steps to train CONCAT to $\text{acc}_{tgt} = 0.95$, then train other grammars for the same number of steps, then report acc_{train} for each of these other grammars. Each result is an averaged over 3 runs.

In addition, we experimented in this table with using other loss functions than cross-entropy: we experiment with adding a Gumbel sampler to the network output, prior to the loss function (GUMB); and adding a stochastic sampler to the network output, and train using REINFORCE (DISCR) (i.e. 'discrete').

Table 18: Example results for measuring compositional inductive bias for end to end architectures, for $n_{att} = 5$, $n_{val} = 10$. ↑ and ↓ denotes columns we want to maximize or minimize respectively.

Repr	Send arch	Recv arch	CONCAT ↑	PERM ↓	ROT ↓	SHUFDET ↑	PAIRSUM ↓
SOFT	RNNAutoReg:dgsend	RNN:dgrecv	0.919	0.859	**0.33**	0.893	**0.61**
	RNNAutoReg:GRU	RNN:GRU	**0.949**	0.94	0.89	**0.976**	0.73
	HierZero:GRU	Hier:GRU	0.85	**0.80**	0.53	0.74	0.63
	RNNZero2L:RNN	RNN2L:RNN	0.870	**0.80**	0.83	0.954	0.66
	RNNZero:RNN	RNN:RNN	0.87	0.829		0.946	0.72
	RNNZero:GRU	RNN:GRU	0.887	0.88	0.891	0.913	0.78
GUMB	RNNAutoReg:dgsend	RNN:dgrecv	0.46	0.38	0.32	0.25	**0.096**
	RNNAutoReg:GRU	RNN:GRU	0.381	0.433	0.388	0.344	0.350
	HierZero:GRU	Hier:GRU	0.36	0.7	**0.26**	0.202	0.27
	RNNZero2L:RNN	RNN2L:RNN	**0.935**	0.28	0.48	0.82	0.492
	RNNZero:RNN	RNN:RNN	0.80	**0.25**	0.501	0.56	
	RNNZero:GRU	RNN:GRU	0.909	0.85	0.746	**0.89**	0.51
DISCR	RNNAutoReg:dgsend	RNN:dgrecv	0.18	**0.30**	**0.262**	0.27	**0.17**
	RNNAutoReg:GRU	RNN:GRU	0.36	0.68	0.34	0.38	0.50
	HierZero:GRU	Hier:GRU	0.40	0.34	0.31	0.34	0.36
	RNNZero2L:RNN	RNN2L:RNN	0.72	0.56	0.42	0.28	0.38
	RNNZero:RNN	RNN:RNN	**0.827**	0.750		0.279	0.282
	RNNZero:GRU	RNN:GRU	0.58	0.80	0.445	**0.61**	0.577

Table 18 shows additional results for end to end training from models first pre-trained supervised on specific grammars. The methodology used to generate these tables was:

- train the sender and a receiver model supervised until they achieve acc_{tgt} on $\mathcal{G}$
- place the sender and receiver end to end, as an auto-encoder
- train the auto-encoder end-to-end for T steps
- measure the accuracy of either the sender or the receiver model on the original language $\mathcal{G}$

The results are depicted in Table 18, using $T = 10,000$. Results are each averaged over three runs.

H SHUF AND SHUFDET GRAMMAR AND ADDITIONAL RESULTS

.

We would like to encourage the models to emerge relocatable atomic groups of tokens, that is something similar to words in natural language. We thus create two artificial grammars, which we would like neural models to acquire quickly: SHUF and SHUFDET. SHUF ('shuffle') permutes the order sub-messages w, prior to concatenation. The permutation is sampled uniformly once per utterance in the language. For example, if object (red,box) maps to utterance'adaaccad', then after permutation of the word order, the shuffled message could be 'ccadadaa', equivalent to 'boxred' in English. SHUFDET ('shuffle deterministically') samples one permutation for each value of the first attribute of the utterance. Thus the permutation is deterministic, given the value of the first attribute, and the sampled permutations.

Table 19: Comparison of various RNN models, searching for low SHUFDET. Results are mean over 10 runs.

Model	Params	PERM	PROJ	ROT	SHUFDET	HOL
LSTM	139909	1.08	2.4	6.3	1.8	> 20
LSTM-Z	139141	**1.15**	2.3	5.4	1.83	> 20
dgsend	106117	0.92	2.3	5.5	1.52	> 20
dgsend-Z	105349	0.92	2.1	4.6	1.54	> 20
HUSendA:RNN	64262	0.92	2.4	10	1.9	> 20
HUSendZ:RNN	40710	0.93	**2.5**	**15**	1.68	> 20
HUSendA:dgsend	178566	1.09	1.9	5.8	1.4	> 20
HUSendZ:dgsend	138758	1.03	1.67	4.5	**1.27**	> 20

In natural language, whilst it is not the case that all sentences can be permuted without changing the meaning, it is the case that many sentences can be re-arranged, without much affecting a human's understanding.

For a Sender, evaluating on SHUF is not reasonable, since there is no obvious way for the Sender to know which order we are evaluating on. Hence, SHUFDET might be reasonable for a Sender model. In natural language, some word orders are dependent on the values of certain words. For example, in French, the adjective 'neuve' follows a noun, whereas 'nouvelle' precedes it.

SHUF and SHUFDET contrast with the other artificial grammars we propose in that we feel that models with a similar compositional inductive bias to humans should acquire these grammars quickly.

Table 19 shows more complete results for our comparison of SHUFDET bias across various recurrent neural architectures.

I END-TO-END TRAINING

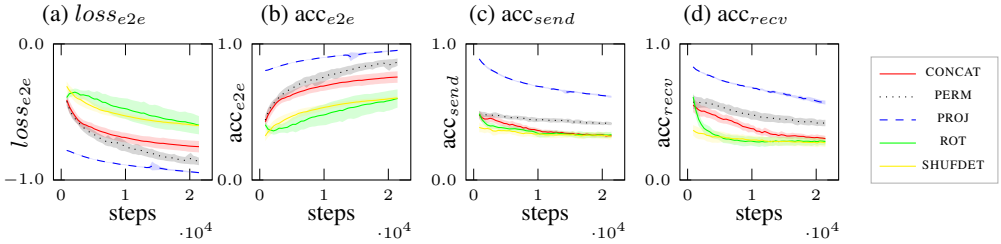

Figure 7: Training curves for LSTM sender and LSTM receiver placed end-to-end after supervised training on the specified grammar. Each curve is mean over 10 runs, and shading is CI95. $n_{att} = 5$.

We experimented with training a Sender and Receiver model supervised on a specific grammar, placing end-to-end, and continuing training, using REINFORCE. Figure 7 shows the results for an LSTM Sender and Receiver. We see clear differences between the grammars, but some are surprising. We expected that CONCAT and PERM would have the smallest $loss_{e2e}$ and the best acc_{e2e}, but PROJ did better, and PERM did better than CONCAT. acc_{send} and acc_{recv} measures the accuracy of the emergent language w.r.t. the original grammar. We thought that CONCAT and PERM would deviate least, but PROJ deviated the least, for reasons unclear. We feel that this scenario might provide opportunities to investigate generalization and exploration under controlled conditions.

J ACQUISITION ACCURACY GIVEN FIXED TRAINING BUDGET

Table 3 is conveniently intuitive to read, however the number of steps to reach convergence is unbounded, and some combinations of model and grammar might never converge. We worked around this issue by stopping training at $b = 20$. An alternative approach is to train for a fixed number of training steps, and report the resulting accuracy. For each model, we train CONCAT until acc_{tgt}, and then train other grammars for the same number of steps. Table 20 shows results for some of the architectures from Table 3. An obvious downside is that we cannot tell which grammars will

Table 20: Compositional inductive bias estimated by accuracy after training for fixed number of steps. In red, results below 0.5, and in green, results over 0.8. acc_{tgt} for CONCAT is 0.99. Results are averaged over 5 seeds.

Model	Params	PERM	PROJ	ROT	SHUFDET	HOL
1-layer MLP	5100	0.995	0.848	0.478	0.741	0.258
2-layer MLP	19428	0.995	0.78	0.481	0.715	0.252
1-layer LSTM	139909	0.967	0.823	0.721	0.894	0.251
2-layer LSTM	272005	0.961	0.817	0.705	0.9	0.249

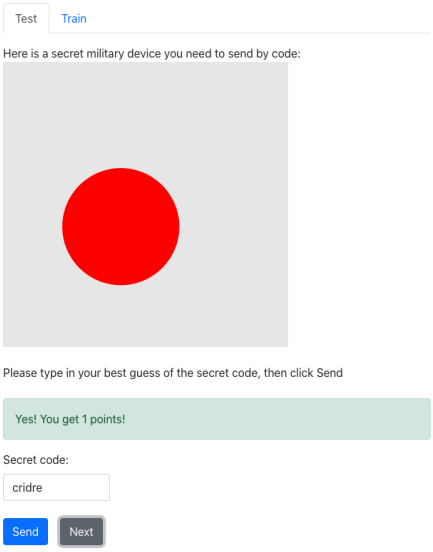

Figure 8: User interface for our 'Secret Spy Codes' game. This image depicts a PERM grammar, where the original utterance was 'redcir'.

ever converge. However, the measures are squashed conveniently into $[0, 1]$, and the experiments are much faster to run (constant time given convergence time of CONCAT). We can see that the relative accuracies correlate well with the results in Table 3.

K HUMAN EVALUATION

We constructed an MTurk (Crowston, 2012) task, 'Secret Spy Codes', in order to evaluate human performance on ICY grammars. Figure 8 shows the user interface. Human subjects are tasked with writing out the code that represents depicted geometric objects. A challenge we found with human experiments was that humans need substantial effort in order to learn just a few new words. Thus, we use objects with only two attributes: shape and color. We considered two scenarios, which we depict as 'eng' and 'synth'.

K.1 DATASET

'synth' uses artificially generated random words for each attribute value. We sample 2-letter words from a vocabulary size of 4. Each utterance therefore has 4 letters: 2 for each of shape and color. Empirically, humans found these words challenging to remember, so we used just three possible values for each attribute. Thus, there were 9 combinations in total.

'eng' uses 3-letter English abbreviations for attribute values, e.g. 'tri' for 'triangle', and 'grn' for 'green'. The words describing each attribute value in 'eng' are relatively easy for a human to learn. Therefore, we used 5 attribute values for each attribute, giving 25 possible combinations.

Subjects had access to a 'training' panel, where they could cycle through example images and utterances for the current grammar, then switch to a 'test' panel to enter their answer. Thus, subjects could obtain a perfect score, given sufficient time to browse through the training examples. However, we hoped that the time required to browse through the training examples would vary depending on how easy the grammar was to memorize.

We held out three color-shape combinations, that were not made available in the training panel. For example, subjects might have access to a red circle and a blue triangle, but not a red triangle. Subjects who could perceive the compositional structure of a grammar should be able to get these holdout instances correct.

K.2 PAYMENT, INCENTIVES AND CURRICULUM

We paid subjects to play games comprising 50 examples, where each game uses single grammar instance. The game provided immediate points rewards, and sound effects for right and wrong answers. We received feedback such as 'good', 'an interesting task', and 'It is very interesting task, kindly upload more tasks like this in future', which suggested that subjects enjoyed playing.

Payment was a base rate plus a linear function of the subject's total score. We found that paying only a base rate worked well initially, but as we ran more tasks, subjects quickly learned to just put random utterances for each example, completing quickly, and scoring 0. Paying a linear function of the subject's total score solved this issue. We paid a base rate in order that some of the harder tasks were not too discouraging.

To avoid overwhelming subjects with learning many new utterances at the start, we start the game with only two color-shape combinations, and add one additional combination every 8 test examples. Subjects have buttons to add and remove color-shape combinations, so they can control their own curriculum. To incentivize subjects to increase the number of color-shape combinations, the score for each example is linearly proportional to the number of color-shape combinations available.

K.3 SCORING

Subjects were given points for each correct answer. The points for each example was calculated as (number available objects at that time) - 1. For ENG dataset, the maximum possible score is this $(25-3-1)*50 = 1050$ (we remove 3 objects for holdout; holdout itself scores identically to other examples), while for SYNTH dataset, the maximum possible score is $(9-3-1)*50 = 250$.

If someone uses the default curriculum, without modifying the number of available cards, then the maximum possible score is $1*8+2*8+3*8+4*8+5*8+10*6 = 180$, independent of dataset.

K.4 ACCEPTANCE CRITERIA

We automatically paid all workers who earned at least a score of 100. We automatically rejected payment to all workers who scored 0. In between these two values, we inspected manually. Anyone who gave the same answer, or almost the same answer, for all examples, we rejected payment for, otherwise we accepted. We noticed that the typical score for anyone putting the same answer for all examples was around 41, which corresponded to the score at chance in this scenario.

For our results tables, we include everyone who scored above 50, and ignore results for anyone who scored below 50.

K.5 EVALUATION

We measured subjects performance in two ways: across all test examples, and uniquely on the 3 held out examples.

Tables 21 and 22 show the results. Analysis of $\text{acc}_{holdout}$ is already included in the main paper body. As far as score and timings, the subjects always have access to a 'training' tab, where they can view the code for all objects except the holdout objects, therefore it is possible to obtain a perfect score in all scenarios, by referring to the training objects. We decided that it was better to provide a

Grammar	N	t (mins)	score	$\text{acc}_{holdout}$
COMP	18	17 ± 4	140 ± 10	0.2 ± 0.1
PERM	17	22 ± 7	160 ± 10	0.04 ± 0.05
PROJ	15	16 ± 3	140 ± 10	0.02 ± 0.04
ROT	18	21 ± 6	140 ± 10	0.04 ± 0.07
SHUFDET	17	19 ± 5	140 ± 10	0.06 ± 0.06

Table 21: Human evaluation results, SYNTH dataset

Grammar	N	t (mins)	score	$\text{acc}_{holdout}$
COMP	15	14 ± 3	200 ± 100	0.6 ± 0.2
PERM	18	30 ± 10	300 ± 100	0.2 ± 0.2
PROJ	17	19 ± 4	160 ± 40	0.04 ± 0.08
ROT	21	24 ± 5	170 ± 30	0.02 ± 0.03
SHUFDET	17	20 ± 8	240 ± 80	0.7 ± 0.2

Table 22: Human evaluation results, ENG dataset

self-service training table, than to simply be measuring who had decided to write down a translation table between object and code, e.g. on a piece of paper. However, both our provision of a training table, and the possibility that subjects write down a translation table, means that there is a negligible difference in scores across grammars, on the training data.