On the Realization of Compositionality in Neural Networks

Baan, J.; Leible, J.; Nikolaus, M.; Rau, D.; Ulmer, D.; Baumgärtner, T.; Hupkes, D.; Bruni, E.

DOI
10.18653/v1/W19-4814

Publication date
2019

Document Version
Final published version

Published in
The BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP at ACL 2019

On the Realization of Compositionality in Neural Networks

Joris Baan[1], Jana Leible[1], Mitja Nikolaus[2], David Rau[1], Dennis Ulmer[1],
Tim Baumgärtner[1], Dieuwke Hupkes[1,*] and Elia Bruni[3,*]

{joris.baan,jana.leible,david.rau,dennis.ulmer}@student.uva.nl
mitja.nikolaus@posteo.de
baumgaertner.t@gmail.com
d.hupkes@uva.nl
elia.bruni@gmail.com

[1]University of Amsterdam, [2]University of Tübingen, [3]Universitat Pompeu Fabra

Abstract

We present a detailed comparison of two types of sequence to sequence models trained to conduct a compositional task. The models are architecturally identical at inference time, but differ in the way that they are trained: our *baseline* model is trained with a task-success signal only, while the other model receives additional supervision on its attention mechanism (Attentive Guidance), which has shown to be an effective method for encouraging more compositional solutions (Hupkes et al., 2019). We first confirm that the models with attentive guidance indeed infer more compositional solutions than the baseline, by training them on the lookup table task presented by Liška et al. (2019). We then do an in-depth analysis of the *structural* differences between the two model types, focusing in particular on the organisation of the parameter space and the hidden layer activations and find noticeable differences in both these aspects. Guided networks focus more on the components of the input rather than the sequence as a whole and develop small functional groups of neurons with specific purposes that use their gates more selectively. Results from parameter heat maps, component swapping and graph analysis also indicate that guided networks exhibit a more modular structure with a small number of specialized, strongly connected neurons.

1 Introduction

Sequence to sequence models (seq2seqs), a subset of neural networks that use sequences as input and output, have enjoyed great success in many NLP tasks such as machine translation (Bahdanau et al., 2015) and speech recognition (Graves et al., 2013). Even though these feats indicate excellent generalization capabilities, the way seq2seqs generalize has found to be different from how humans do. In particular, seq2seqs lack of compositional understanding: the ability to construct new representations by combining familiar primitive components (e.g. Szabó, 2012). Humans, instead, heavily rely on compositionality to learn complex functional structure efficiently (Schulz et al., 2016). Once the primitive components are understood, a possibly infinite amount of novel combinations can be made, which allows for large scale generalization from a limited amount of examples (Fodor, 1975). For instance, sentences consist of words, which in turn consist of characters constructed from strokes.

Recently, Liška et al. (2019) have shown how seq2seqs can produce many different fits on the training data using stochastic gradient descent, but rarely, if ever, find a compositional solution. The authors introduce a new data set called the **lookup table** task, which tests for out of distribution generalization. This data set will be discussed in more detail in Section 2.1.

As a remedy, Hupkes et al. (2019) proposed Attentive Guidance (AG), a training technique which encourages seq2seqs to encode a more compositional solution without changing their internal architecture. AG provides additional information about the structure of the input sequence by supervising the attention mechanism of a model. As a result, the model is able to find what are the basic components of the lookup table task and how to combine them in a compositional manner.

Thanks to this work, we are now in the unique position of having a compositional (from now on *AG*) and non-compositional (from now on *base-*

[*] Shared senior authorship

Proceedings of the Second BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP, pages 127–137
Florence, Italy, August 1, 2019.

line) model that have identical architectures, but implement very different approaches to the same task. In this paper, we compare those two models and aim to find structural differences between the way they organise their weights and form their representations, that could be indicative of compositional solutions. In particular:

- We show, through inspection of the parameter space and activations, that individual neurons in the AG model show a degree of specialization with respect to specific inputs that is unseen in baseline models.
- We demonstrate, by substituting parts of both models with the corresponding component of its counterpart, which model sections contribute most to the observed compositional behavior in AG models.

These differences confirm the findings of Hupkes et al. (2019) that seq2seqs do not necessarily require big architectural adjustments to handle compositionality, since a network with identical architecture is capable of finding such a solution. Furthermore, these findings could be exploited to inform architectural changes in models, such that their priors to infer compositional solutions increase even when they are not provided explicit additional feedback on the compositional structure of the data.

2 Setup

In our experiments, we compare vanilla seq2seq with models that are trained with AG. Below, we briefly discuss both setups and the data we use for our experiments.

2.1 Task

For our experiments, we use the lookup table composition task proposed by Liška et al. (2019), which was created to test the compositional abilities of neural networks. In this task, atomic lookup tables are created as to define a unique mapping from one binary string to another binary string of the same length. These atomic tables are then applied sequentially to a binary input string and yield a binary string. To give an example: if $t1(001) = 110$ and $t2(110) = 001$, then the function $(t1 \circ t2)(001) = 001$ can be computed as a composition of $t1$ and $t2$. See Table 1 for a more comprehensive example.

Following Hupkes et al. (2019), we generate eight atomic lookup tables with strings of length 3 and use them to produce all 64 possible length two compositions. This forms the basis of the dataset that all experiments were performed on. To test the model's ability of generalization on a more granular level, we compose four test sets with an increasing level of difficulty. For the first test set, we remove 2 out of 8 inputs for every composition (*heldout inputs*). For the second and third testset, we remove 8 random table compositions from the training set (*heldout compositions*), as well as all compositions that either contain $t7$ or $t8$ (*heldout tables*). Finally, we create a test set by removing all compositions that contain a combination of tables $t7$ and $t8$ from the training set (*new compositions*). The nature of the tasks requires the models to make use of the underlying compositionality. If this structure is not exploited, it is impossible to reliably find the correct solutions for the test data. For more details, we refer to Liška et al. (2019) and Hupkes et al. (2019).

Atomic	Atomic	Composed
$t1$	$t2$	$t1 \circ t2$
$000 \rightarrow 111$	$000 \rightarrow 100$	$000 \rightarrow 011$
$001 \rightarrow 010$	$001 \rightarrow 101$	$001 \rightarrow 110$
$010 \rightarrow 101$	$010 \rightarrow 110$	$010 \rightarrow 100$
$\dots$	$\dots$	$\dots$

Table 1: Example for atomic lookup tables ($t1$ and $t2$) of length 3 and a composition of length 2 ($t1 \circ t2$).

2.2 Baseline

The baseline model consists of an encoder-decoder architecture with an attention mechanism (Bahdanau et al., 2015) and Gated Recurrent Units (GRU)[1] (Cho et al., 2014).

GRUs compute the hidden activations h_t based on the previous hidden state h_{t-1} and the representation of the current input x_t in the following way (biases were omitted for clarity):

$$
\begin{aligned}
z_t &= \sigma(W_{iz}x_t + W_{hz}h_{t-1}) \\
r_t &= \sigma(W_{ir}x_t + W_{hr}h_{t-1}) \\
\tilde{h}_t &= \tanh(W_{ih}x_t + W_{hh}(r_t \circ h_{t-1})) \\
h_t &= (1 - z_t) \cdot h_{t-1} + z_t \cdot \tilde{h}_t,
\end{aligned}
$$

[1]We also trained models with Long-Short Term Memory units (Hochreiter and Schmidhuber, 1997) but found the results to be very similar and therefore decided to omit the latter from this work.

where we call z_t and r_t the activations of the *update gate* and *reset gate*, respectively.

2.3 Attentive Guidance

The AG model used in this work is identical to the baseline model in terms of architecture. The only difference occurs during the training procedure, where an additional loss term is enforced on the weights of the attention mechanism at decoding time step t for input token i, $\hat{a}_{i,t}$:

$$\mathcal{L}_{\text{AG}} = \frac{1}{T}\Big(\sum_{t=1}^{T}\sum_{i=1}^{N} -a_{i,t}\log \hat{a}_{i,t}\Big),$$

where $\hat{a}_{i,t}$ denotes the target attention weights. The attention loss is computed with an additional set of labels, that express how the input should be segmented and in which order it should be processed. Hupkes et al. (2019) show that providing this additional supervision consistently improves the solutions found for the lookup table task: the guided models were found to have perfect generalization capabilities on the *heldout compositions* and *heldout inputs* and also perform well on *heldout tables* and *new compositions*. As inputs are supposed to be processed sequentially in our case, the target attention pattern is strictly monotonic, i.e. the target attention weights over the sequence are realized in a diagonal matrix.

2.4 Experiments

We train five baseline and AG models with the same hyperparameters and the Adam optimizer (Kingma and Ba, 2015). Given the small vocabulary, we use an embedding size of 16 and a hidden size to 512. All models were trained for a maximum of 100 epochs with an attention mechanism, determining attention weights by using a multi-layer perceptron. Models were selected by their best accuracy on a held-out set. A comprehensive list of model performances on the different sets can be found in the Appendix. The model implementations themselves stem from the `i-machine-think` codebase.[2]

In the following, we perform three different suits of experiments. Firstly, we examine the parameter space of both models (Section 3). Secondly, we take a closer look at the activations of single neurons and the GRU gates (Section 4). Lastly, in Section 5, we perform two different ablation studies: we make components of one model interacting with the components of the other and we distill the network via strongly connected neurons.

[2]Available under https://github.com/i-machine-think/machine.

3 Inspecting the Parameter Space

In this section, we look at the parameter space of the baseline and AG models. All discoveries regarding the parameter space were validated by comparing 5 runs of the same model class to make sure that observed differences can be ascribed to the differences in models and not different weight initializations.

3.1 Weight Inspection

To gain a better understanding of the organization of the weights, we generated weight heat maps with the y-axis representing the weights going from all neurons to one neuron of the next layer (incoming weights) and the x-axis the weights going from one neuron to all neurons of the next layer (outgoing weights). Neural networks are known to be good at distributing their weights rather than have strong spatial organization, which makes it interesting to see whether such heat maps would reveal any differences in the organization of weights between AG and baseline models [3].

The most striking difference between AG and baseline arises for the decoder embedding, as can be seen in Figure 1. The baseline model exhibits small weights whereas the AG model shows bigger weights in rows 2-10. Row number two is an exception, since it is equally strong for both networks. This might be explained by the fact that this row represents the start-of-sequence (SOS) token, which could be sending a stronger error signal for both models.

3.2 Neural Connectivity

Since the heat maps of the weight matrices for larger layers were hard to interpret, we explored a more intuitive visualization of the network's parameter space. We took neurons as nodes, and weights between neurons as edges. The thickness and color of an edge represents the magni-

[3]Note that we do not normalize reported weights or activations by the activity of the 'pre-synaptic' neurons connected to it. This would be interesting to explore in future research, since a neuron's activation and the importance of its weight is in part dependant on the mean activation of its predecessors.

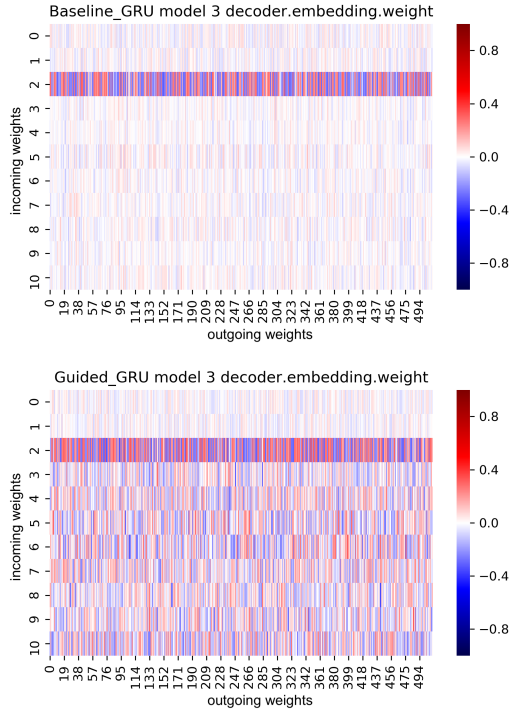
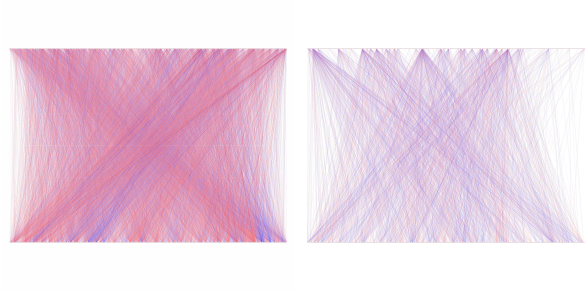
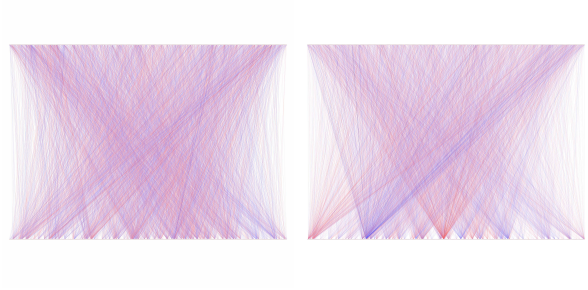


Figure 1: Heatmap of the decoder embedding weight values. Outgoing weights correspond to weights going from the embedding to one decoder output neuron, and incoming weights to all weights going from the embedding to all decoder output neurons. (Best viewed in color)



Left: baseline. Right: AG. Encoder update gates W_{hz}.



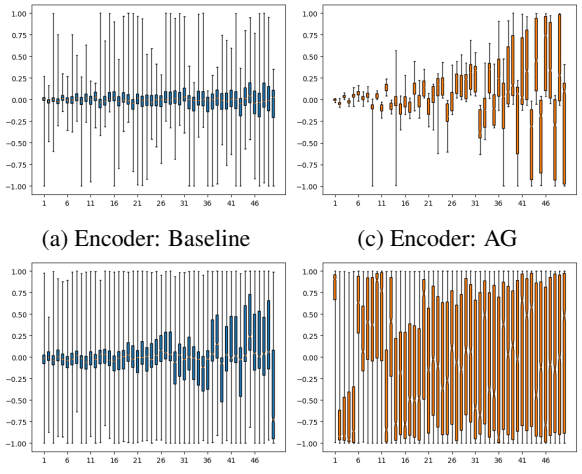
Left: baseline. Right: AG. Decoder update gates W_{iz}.

Figure 2: Visualization of weight matrices W_{hz} of the encoder and W_{iz} of the decoder. Weights going from the previous to the next layer are represented by lines going from bottom to the top. The color reflects the weight value, where blue denotes negative, red positive and white zero. (Best viewed in color)

tude of the weight. To prevent clutter, we applied thresholding to remove edges that corresponded to weak weights. For the encoder and decoder, we used a threshold of ± 0.2 and ± 0.17 respectively, which corresponds on average (between AG and vanilla models) to the strongest one percent of the weights.

The goal is to understand how the parameter space is structured and to see whether any differences between AG and baseline models can be found, for example, because of a stronger modularity, grouping or specialization of neurons in AG models.

Figure 2 depicts the update gate weights W_{hz} of the encoder on the top and the weights W_{iz} of the decoder at the bottom. The weights of the previous layer to the next are represented by edges going from bottom to top. The most striking difference is that the baseline weights seem much more cluttered, whereas the AG model exhibits a few distinct, strongly polar neurons - neurons whose weights are on average negative or positive. Neurons that have many strong connections occur in the top layer of the encoder of the AG model in



(a) Encoder: Baseline (c) Encoder: AG

(b) Decoder: Baseline (d) Decoder: AG

Figure 3: Distributions of activation values for 50 randomly sampled neurons for baseline (blue) and AG (orange) for both encoders (top) and decoders (bottom). Whiskers show the full range of the distribution. (Best viewed in color)

W_{hz} and W_{hr}. Similarly, strong connected neurons can be found at the bottom layer of the AG's decoder in W_{iz} and W_{ir}. The finding of highly connected neurons seems to further reinforce the hypothesis that AG models learn to specialize using fewer but more strongly connected neurons, which could help to learn a more modular solution.

Another interesting phenomenon that holds for both models can be observed by looking at the difference between update and reset gates of the same network (not shown here for the sake of space): The polarity of the neurons that are on average strongly positive or negative are inversely related. A possible explanation for this is that, when information from the current hidden state is to be retained, that same part is being reset in the previous hidden state.

4 Analyzing Activations

While analyzing the model weights gives us insight into the general trained structure of the model, analyzing activations lets us examine how the different model types respond to certain inputs. We thus try to identify groups of neurons that specialize to respond to certain inputs and provide further inside into the GRU's gating behavior.

4.1 Functional Groups

We hypothesize that solving the task compositionally is done by distinct groups of neurons in the network. Each group addresses different functionalities. For example, a group of units in the encoder could be responsible for representing the presence of the current table in the input sequence, as proposed in the previous section.

An indicator for this behavior can be seen in Figure 3, where we sampled 50 random neurons from the encoder and decoder of both models and tracked their activation values emitted over the samples in the test set. We can see that in contrast to the baseline, some neurons of the AG only produce activations in specific value ranges, which could be a hint for a potential specialization. The same can be found inside the AG's decoder, although most of the neurons sampled seem to cover the whole value range during processing.

To test this hypothesis, we analyze which hidden activations are crucial for correctly predicting the current table at a time step. The baseline model is expected to not be able to predict the presence of single tables because it fails to see the tables as parts of a compositional task and instead memorizes the combinations it has encountered during training.

In a first experiment, we use diagnostic classification (DC, Hupkes et al., 2018), which consists in training linear classifiers on the hidden activations to predict a certain feature. In this case, we use the encoder's activations to predict the table in the input sequence of the corresponding time step. For example, if the input was '000 t1 t2', we trained the classifier to predict 't1' for the encoder activations of the second time step and to predict 't2' for the activations of the third time step. Similarly to the methodology of Dalvi et al. (2019), we subsequently added units to a set, depending on the absolute weight they were assigned in the diagnostic classifier.[4] After each addition, we re-calculated the accuracy for the prediction. This process was repeated until 95 % of the overall accuracy (with all units) is reached, the resulting subset of units forms the *functional group*.[5]

The results are shown in the first row of Table 2. All numbers are averaged over the five trained models. Some differences arise in the functional group size of the models: While for the baseline models on average 35 units are required to make a good prediction, the information is stored in only 2 units in the guided models.

To verify whether the units in the functional group are actually important units in the model, we further analyzed the strengths of the weights connected to each of the units. On average, 93% of the units in the functional group of the AG models can be found in the top 5% of the units with the strongest absolute weight values. We conclude that the units of the functional group are highly connected and thus very likely to play an essential role in the functionality of the model.

Assuming that the information of the current table being stored in the encoder activations is used by the decoder to perform according calculations, we expect that by using the gate activations of the

[4]However, unlike Dalvi et al. (2019), we do not use any regularization on the DC to contrast the different degrees to which information is distributed across neurons in the two model types.

[5]Applying the methods development by Lundberg and Lee (2017) seems to confirm the responsible neurons we found, but selects more neurons and gives less consistent results, which we trace back to the extensive approximations required and some model assumptions (e.g. feature independence) being violated.

In	Model	Accuracy	#Units
h_t^{enc}	BL	.93 (.98)	35
	AG	.98 (1.)	2
z_t^{dec}	BL	.51 (.53)	52
	AG	.96 (1.)	22.2
r_t^{dec}	BL	.50 (.52)	44
	AG	.96 (1.)	20.8

Table 2: Performance of diagnostic classifiers for predicting the current input table with the hidden activations (h_t^{enc}) of the encoder, the input gate activations (z_t^{dec}) or the reset gate activations (r_t^{dec}) of the decoder of the baseline (BL) and Attentive Guidance (AG) model. The third column shows the accuracy when predicting using the functional group of units and in brackets the accuracy when using all units. The fourth column displays the average number of units in the functional group across different runs (which can be either hidden units or gate activations).

decoder it is also possible to predict the current input table. We use the same methodology as in the previous experiment, with the only difference that the inputs for the diagnostic classifier are the activations of the decoder gates. Results are shown in the second and third rows of Table 2. Using all gate activations of the update or the reset gate of GRUs, we are able to perfectly predict the current table in the guided models. With the baseline model, an accuracy of only around50 % is reached.[6] The size of the functional groups in the guided models is remarkably larger than with the encoder hidden activations, showing that the information is more distributed over the gates. This difference can be explained by the fact that the gates are not mainly representing information, but using represented information to perform calculations. Further, distribution of information across the gates is more likely because a gate activation affects only one hidden unit while a hidden layer activation can possibly affect all gates in the upcoming time step (Hupkes and Zuidema, 2017).

In another experiment, we aim to predict the current time step with the activations of the encoder.[7] We assume that counting is an essential part of solving the task in a compositional manner. The methodology is the same as in the previously described experiments. The result pattern (cf. Table 3) can be compared to the first experiment: Using all units, it is possible to predict the time step with all models, but, in the guided attention models, the information is more concentrated in functional groups than units.

In	Model	Accuracy	#Units
h_t^{enc}	BL	.95 (.98)	40
	AG	1.0 (1.0)	2

Table 3: Performance of diagnostic classifiers for predicting the current time step with the hidden activations (h_t) of the encoder. The second column shows the accuracy when predicting using the functional group of units and in brackets the accuracy when using all units. The third column displays the number of units in the functional group.

These results, implying that some neurons specialize in tracking the current time step and reacting to distinct inputs, demonstrate that the AG model uses information about the current table in the decoder to perform operations in a compositional way (treating the tables as distinct parts). The baseline model does not show distinct activation patterns in the gates for specific tables.

4.2 Gating behavior

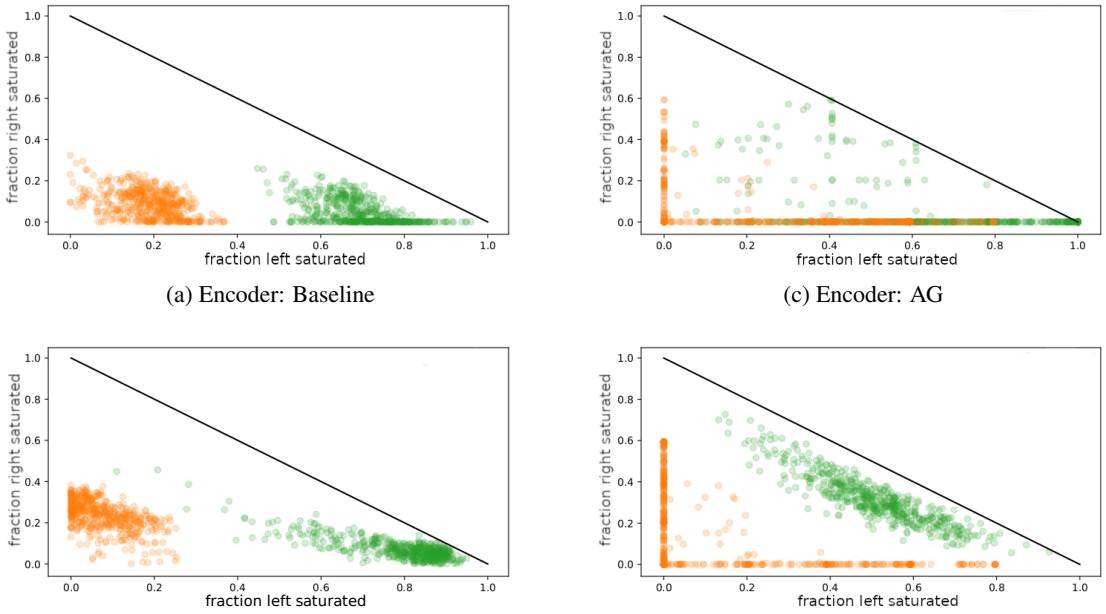
Based on the findings in previous section, we also expect the usage of reset and update gate to be significantly different for the two models under scrutiny. To study this, we use a technique introduced by Karpathy et al. (2015), that considers, for each gate in the network, the fraction of samples for which it is left-saturated (activation smaller than 0.1) or right-saturated (activation greater than 0.9), where being left-saturated corresponds to being closed and right-saturated to being open.

We show the results in Figure 4. The plots reveal a clear difference between the usage of gates in the baseline and the guided models. The guided models seem to be more distinct in their gate activation: Activations tend to stick to the axes. Values close to the diagonal mean that the respective gate is mostly saturated, values close to the axes reveal gates that are either saturated to one side or not saturated. Values in between, mostly seen in the baseline models, indicate gates that are rarely saturated.

The activation pattern of the update gates shows clear differences between the baseline and the AG models. In the baseline, they are mostly left saturated, which means that new information is rarely

[6] Accuracy with a majority classifier for the task is 12.5 %.

[7] For example, if the input was '000 t1 t2', we trained the classifier to predict '0' for the encoder activations of the first time step, '1' for the encoder activations of the second time step and '2' for the activations of the third time step.



(a) Encoder: Baseline

(c) Encoder: AG

(b) Decoder: Baseline

(d) Decoder: AG

Figure 4: Gate activation plots for reset gate r_t and update gate z_t. (Best viewed in color)

incorporated in the calculations, in both the encoder and decoder. In the encoder of the AG model, most gates are also only left-saturated, but there is a considerable amount of outliers which could be some units that are highly specialized to specific inputs. In the AG decoder, some gates are also often right-saturated, allowing for the intake of new information. One possible interpretation is that the gates in the AG decoder model selectively allow the relevant input for the current time step to be included in the calculations.

5 Ablation studies

In this section, we first swap components of both models and measure the effects on the models' performances to identify crucial model parts. We then check performance of the AG model when only its strongly connected neurons are used.

5.1 Component substitution

To understand to what extent specific components of a seq2seqs contribute to compositionality, we take components from one trained model and place them into the other model. We freeze the weights of the replaced component to prevent any re-learning, and retrain the resulting model using its original training procedure. We extract a total of eight different components. The entire encoder and decoder as well as embeddings, internal GRU weights for input to hidden (W_{ih}), and recurrent hidden to hidden (W_{hh}) for both encoder and decoder. The experiment is twofold: components taken from a model trained with AG are placed into a baseline model and retrained without AG to examine whether a baseline can still learn a compositional solution. Additionally, components from a baseline model are placed into an AG model which is retrained with AG to examine whether the model can still learn a compositional solution without being able to adjust the parameters of the baseline component. We tune 16 models for each new component by retraining the remaining original parts for a maximum of 100 epochs, with a learning rate of 0.001 to allow for limited adjustment.

When retraining an AG model with a frozen baseline component, we expect the performance to drop when that component is important for a compositional solution, as the network is apparently unable to recover itself. Conversely, if a baseline model with a frozen component extracted from an AG model is retrained without AG and performs better, that component might contain weights organized in such a way that it forces the baseline model to retrain itself in a more compositional manner. Table 4 shows the results of substituting components on *new compositions*, which is considered the hardest compositional task. None

of the baseline models given a frozen component from an AG model are able to retrain themselves such that they significantly increase performance. Thus, we left those results out of Table 4 for the sake of brevity.

For the AG models with frozen baseline components, the encoder embeddings seem irrelevant for a compositional solution: using baseline embeddings result in a similar score. The decoder embeddings, however, do seem to play a role, as indicated by a much lower score than the original AG model. This seems to be in line with the differences in heat maps shown earlier in Figure 1. Replacing the entire encoder results in a 80% drop in accuracy to 0.167. Interestingly, the encoder hidden to hidden (W_{hh}) weight can be replaced without as big a drop, and using the baseline input to hidden (W_{ih}) weights actually improves the accuracy. Finally, replacing the decoder's W_{ih} weights drops the accuracy to around 0.6, but doing the same for the decoder's W_{hh} weights again results in an unexpected increase to almost 0.9. This seems to indicate that the W_{ih} weights of the decoder play an important role in a compositional fit, as the model is unable to recover itself when using baseline decoder W_{ih} weights. The increase in performance after replacing either the encoder's W_{ih} or the decoder's W_{hh} implies that training with AG actually produces suboptimal weights for these components. Perhaps the use of a frozen baseline component in a model retrained with AG acts as some kind of regularization and incentivizes the remaining components of the model to become more compositional. Another explanation could be that the AG loss does not provide an appropriate signal for all components, and should thus not be backpropagated to all of them.

Model	Component	Accuracy (NC)
AG	-	$.82 \pm .12$
AG	Encoder	$.17 \pm .11$
AG	Encoder Emb	$.75 \pm .09$
AG	Encoder W_{ih}	$.89 \pm .05$
AG	Encoder W_{hh}	$.79 \pm .12$
AG	Decoder	$.12 \pm .05$
AG	Decoder Emb	$.31 \pm .07$
AG	Decoder W_{ih}	$.60 \pm .08$
AG	Decoder W_{hh}	$.91 \pm .03$
BL	-	$.01 \pm .02$
BL	Encoder	$.02 \pm .02$
BL	Decoder	$.02 \pm .02$

Table 4: Sequence accuracy on new compositions (NC). Accuracy is averaged over three models and depicted with its standard deviation. The model being retrained is specified the first column, the component taken from the opposite model and frozen specified in the second column.

5.2 Neuron pruning

After showing in Section 4.1 that a few strongly connected neurons organized in functional groups carry out specific functions, we want to exhaust this observation and see if the model can still successfully solve the task by using **only** strongly connected neurons. We remove all weakly connected neurons, keeping only 5% of neurons with the biggest weights of the encoder and decoder of the trained AG models respectively. Distilling the network in this way results in a performance drop to 12.4% sequence accuracy on the new compositions task, averaged over all models. Re-training the network for 20 epochs fully restores the functionality and even yields better performance of 92.5% on average compared to the full network with 82.3%. We retrained the model using the same parameters as in the main training procedure (see Section 2.4).

The loss in performance that occurs when neurons are removed indicates that some functionality is distributed among weakly connected neurons. However, the fact that their functionality can be taken over by other neurons shows that weakly connected neurons do not play a crucial role.

We conclude that most of the neurons do not contribute to a compositional solution at all and therefore only an extremely small subset of all neurons of the AG model suffices to solve the task after retraining. Those neurons exhibit strong weights and are specialized in functional groups. Networks that find a compositional solution seem to rather form a small number of highly specialized neurons than distributing functionality over the whole network.

6 Conclusion

Thanks to Attentive Guidance, seq2seqs are able to generalize compositionally on the lookup table task when, without it, they cannot (Hupkes et al., 2019; Liška et al., 2019). In this paper, we presented an in-depth analysis of the differences

between an attention-based sequence to sequence model trained with and without Attentive Guidance. Any identified differences can contribute to our understanding of what makes seq2seq better compositional learners and help with the design of a new generation of compositional learning architectures.

Our main finding is that guided networks have a more modular structure: small subsets of well-connected neurons are responsible for specific functions. Having specialized neurons could be crucial to a compositional solution. We have also shown via component substitutions how these neurons seem to play a more crucial part in specific model components like the encoder / decoder gates and decoder embeddings, while playing a negligible role in others.

Future research could focus on exploiting the findings about modularity and specialization of neurons to investigate whether similar compositional solutions can be achieved without the explicit use of Attentive Guidance, such as recently shown by Korrel et al. (2019) Additionally, it would be interesting to find out why models with fewer parameters cannot learn to solve the lookup table task (Hupkes et al., 2019), while we know from our distillation experiments that only 26 neurons in the encoder and decoder are needed to implement a perfect solution.

Acknowledgements

DH is funded by the Netherlands Organization for Scientific Research (NWO), through a Gravitation Grant 024.001.006 to the Language in Interaction Consortium. EB is funded by the European Union's Horizon 2020 research and innovation program under the Marie Sklodowska-Curie grant agreement No 790369 (MAGIC).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP. Doha, Qatar*, pages 1724–1734.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, Yonatan Belinkov, Anthony Bau, and James Glass. 2019. What is one grain of sand in the desert? analyzing individual neurons in deep nlp models. In *Proceedings of the AAAI Conference on Artificial Intelligence AAAI. Honolulu, Hawaii, USA*.

Jerry A Fodor. 1975. *The language of thought*, volume 5. Harvard University Press.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE.

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

D. Hupkes and W. Zuidema. 2017. Diagnostic classification and symbolic guidance to understand and improve recurrent neural networks. Interpreting, explaining and visualizing deep learning. *NIPS2017*.

Dieuwke Hupkes, Anand Singh, Kris Korrel, German Kruszewski, and Elia Bruni. 2019. Learning compositionally through attentive guidance. In *20th International Conference on Computational Linguistics and Intelligent Text Processing (CICLing)*.

Dieuwke Hupkes, Sara Veldhoen, and Willem Zuidema. 2018. Visualisation and 'diagnostic classifiers' reveal how recurrent and recursive neural networks process hierarchical structure. *Journal of Artificial Intelligence Research*, 61:907–926.

Andrej Karpathy, Justin Johnson, and Li Fei-Fei. 2015. Visualizing and understanding recurrent networks. In *Proceedings of the International Conference on Learning Representations 2016*, pages 1–13.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR. San Diego, CA, USA*.

Kris Korrel, Dieuwke Hupkes, Verna Dankers, and Elia Bruni. 2019. Transcoding compositionally: using attention to find more generalizable solutions. *BlackboxNLP 2019, ACL*.

Adam Liška, Germán Kruszewski, and Marco Baroni. 2019. Memorize or generalize? searching for a compositional rnn in a haystack. In *Proceedings of AEGAP (FAIM Joint Workshop on Architectures and Evaluation for Generality, Autonomy and Progress in AI)*.

Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, pages 4768–4777.

Eric Schulz, Josh Tenenbaum, David K Duvenaud, Maarten Speekenbrink, and Samuel J Gershman. 2016. Probing the compositionality of intuitive functions. In *Advances in neural information processing systems*, pages 3729–3737.

Zoltan Szabó. 2012. The case for compositionality. *The Oxford handbook of compositionality*, 64:80.

A Model performances

Sequence accuracy for the increasingly difficult tasks heldout compositions (HC), heldout inputs (HI), heldout tables (HT) and new compositions (NC) of the baseline (Tab. 5) and the AG (Tab. 6) models.

Run	HC	HI	HT	NC
1	.25	.20	.04	.00
2	.16	.20	.06	.00
3	.25	.23	.04	.03
4	.22	.23	.03	.03
5	.23	.20	.07	.06

Table 5: **Baseline** GRU models

Run	HC	HI	HT	NC
1	1.0	1.0	0.85	0.69
2	1.0	1.0	0.98	0.97
3	1.0	1.0	0.88	0.81
4	1.0	1.0	0.98	0.97
5	1.0	1.0	0.94	0.91

Table 6: **Guided Attention** GRU models