Can Transformers *Jump Around Right* in Natural Language? Assessing Performance Transfer from SCAN

Rahma Chaabouni[*†] Ecole Normale Superieure
Roberto Dessì Facebook AI & Pompeu Fabra
Eugene Kharitonov[*] Facebook AI
{chaabounirahma, roberto.dessi11}@gmail.com kharitonov@fb.com

Abstract

Despite their failure to solve the compositional SCAN dataset, seq2seq architectures still achieve astonishing success on more practical tasks. This observation pushes us to question the usefulness of SCAN-style compositional generalization in realistic NLP tasks. In this work, we study the benefit that such compositionality brings about to several machine translation tasks. We present several focused modifications of Transformer that greatly improve generalization capabilities on SCAN and select one that remains on par with a vanilla Transformer on a standard machine translation (MT) task. Next, we study its performance in low-resource settings and on a newly introduced distribution-shifted English-French translation task.

Overall, we find that improvements of a SCAN-capable model do not directly transfer to the resource-rich MT setup. In contrast, in the low-resource setup, general modifications lead to an improvement of up to 13.1% BLEU score w.r.t. a vanilla Transformer. Similarly, an improvement of 14% in an accuracy-based metric is achieved in the introduced compositional English-French translation task. This provides experimental evidence that the compositional generalization assessed in SCAN is particularly useful in resource-starved and distribution-shifted scenarios.

1 Introduction

While sequence-to-sequence (seq2seq) models achieve remarkable performance in many tasks (Sutskever et al., 2014; Raffel et al., 2019; Adiwardana et al., 2020), they often fail to generalize in a systematic way (Baroni, 2019; McCoy et al., 2020; Hupkes et al., 2020; Kharitonov and Chaabouni, 2021; Dankers et al., 2021). These shortcomings are particularly obvious in the experiments on the SCAN domain (Lake and Baroni, 2018; Loula et al., 2018; Bastings et al., 2018).

In SCAN, inputs are instructions that describe trajectories and outputs define sequences of actions to follow them (see Table 1). To illustrate how SCAN probes a model for compositional generalization, imagine that we train it on a set of instructions {`jump`, `run`, `turn`, `turn twice`, `run twice`}, but test it on `jump twice`. Strictly speaking, nothing in the training data indicates that the model *must* output `JUMP JUMP` instead of e.g. `JUMP`. However, it is hypothesised that a bias for such compositional, human-like induction is beneficial (Lake and Baroni, 2018; Lake, 2019).

This hypothesis stumbles into a perplexing situation: despite failing at compositional generalization, considered as a core requirement for language understanding, seq2seq models have tremendous success in practice. Is the type of compositional generalization, that SCAN probes for, useful for NLP tasks? If so, in what scenarios?

In this work, we aim to answer this question. Firstly, we introduce focused modifications to Transformer that greatly improve accuracy performance on SCAN. To build such modifications, we exploit two observations: (i) Transformer's architecture is very similar to convolution-based seq2seq models (ConvS2S) (Gehring et al., 2017), (ii) ConvS2S performs well on SCAN (Dessì and Baroni, 2019). This capability (ii) is hypothesized to be due to explicitly localized representation of the sequences, where only deep layers of the model can access more distant tokens (Dessì and Baroni, 2019; Hupkes et al., 2020). Such a capability should also benefit to natural language processing as human languages are proved to favor local syntactic constructions (Futrell et al., 2015). Motivated by these observations, we focus on the major differ-

[*] Equal contribution. [†] work was done while R.C. was at Facebook AI.

Proceedings of the Fourth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP, pages 136–148

ences between ConvS2S and Transformer, namely the localized receptive field span and the gating mechanism, to inject into Transformer inductive biases useful for SCAN.

As a testbed, we use the machine translation (MT) domain as one of the most popular applications for seq2seq models. We consider both resource-rich (IWSLT'14 German→English) and low-resource (Nepali&Sinhala↔English) setups. Finally, to evaluate SCAN-style capabilities in natural language tasks, we build a dataset that probes whether models can systematically generalize w.r.t. noun-adjective ordering while translating from English to French. We construct this dataset by filtering the EuroParl corpus (a part of WMT'14).

Our results indicate that combining two ConvS2S-inspired changes improves accuracy on one SCAN split (SCAN-jump) from 3.4% to 43.0%, while maintaining a high accuracy on the other splits (SCAN-simple and SCAN-around-right). As expected, given that SCAN is an artificial diagnostic dataset, not all modifications lead to equal improvements on an MT task. We select one of the considered modifications that performs on par with the vanilla Transformer on IWSLT'14.

Testing the selected modification on low-resource data, we observe that it provides between 3.6% and 13.1% BLEU improvements over Transformer. On the noun-adjective ordering dataset, we find that our modification results into gains in generalization of 14%.

This leads to the following picture: the localized attention, augmented by a gating mechanism, provides a useful inductive bias that proves to be beneficial for SCAN-style generalization. Additionally, it turns out useful in low-resource and distribution-shifted settings. Thus, testing seq2seq models on SCAN while controlling for a non-degraded performance leads to improvement in domains where syntactic compositionality is crucial for a task success.

2 Transformers and ConvS2S

Architecture overview Both Transformer and ConvS2S are encoder-decoder architectures (Sutskever et al., 2014), where the decoder has an attention mechanism to peek into the encoder's representation (Bahdanau et al., 2014). These representations are obtained by embedding the inputs, adding a positional embedding, and passing them through a sequence of layers.

In Transformer's encoder, the output representations are the result of a sequential application of two (sub)layer types: self-attention and fully connected layers. The input representation can "skip" any sublayer via a residual connection. The output of the sublayer is passed through a dropout mechanism and added to the residual. This sum is then layer-normalized. Any relation between input tokens is modeled solely by self-attention modules.

In ConvS2S, the encoder is also a sequence of identical blocks. The inter-term dependencies are modeled by 1D convolutions with GLU activation functions (Dauphin et al., 2017).[1] In contrast to self-attention, convolutions have a finite kernel size, thus effectively capping the maximal distance of intermediate dependencies that can be modeled. The GLU activation function serves as a gate, allowing ConvS2S to control the balance between residuals and the output of the convolution. After the GLU operation, the intermediate representation is added to the residual and scaled. The output of the final convolution is then passed into a fully connected layer. In ConvS2S and Transformer, decoders have similar structures to those of encoders, with an additional decoder→encoder attention layer after the convolution and self-attention blocks, respectively.

Despite the similarities, there are numerous low-level differences between the two architectures: normalization (layer norm (Ba et al., 2016) vs. weight normalization (Salimans and Kingma, 2016)), optimization (Adam (Kingma and Ba, 2014) with a ramp-up vs. NAG (Sutskever et al., 2013)), etc. *A priori*, any of those can affect models' inductive biases. However, we concentrate on some of the most obvious architectural differences: the limited convolution span and GLU activations.

We believe these features can greatly affect models' performance on SCAN. Indeed, SCAN has only local dependencies between tokens, thus the ability to avoid spurious correlations with more distant tokens can be useful. Similarly, the ability to weight contributions from the token interactions into the intermediate representation is intuitively prerequisite to build compositional representation.

GLU (Dauphin et al., 2017) Given a vector input $\boldsymbol{x}$, GLU splits it in two equally sized halves $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$; one is passed through a sigmoid ($\sigma(x) = (1 + e^{-x})^{-1}$). Then both parts are pointwise multiplied:

[1] GLU was introduced as a combination of a convolution network and an activation; we follow the Pytorch convention and consider it as separate blocks for convenience.

jump	⇒	JUMP
jump around right	⇒	RTURN JUMP RTURN JUMP RTURN JUMP RTURN JUMP
turn left twice	⇒	LTURN LTURN
jump opposite left after walk around left	⇒	LTURN WALK LTURN WALK LTURN WALK LTURN WALK LTURN LTURN JUMP

Table 1: Examples of SCAN trajectories and instructions, adopted from (Lake and Baroni, 2018).

$$GLU(\boldsymbol{x}) \coloneqq \sigma(\boldsymbol{x}_1) \odot \boldsymbol{x}_2 \tag{1}$$

This allows a network to implement a gating mechanism, where one half of its output gates the signal from the second.

Self-attention Instead of convolutions, Transformer uses multi-headed self-attention to model interactions between tokens. Given n embeddings, $\boldsymbol{x}_1, \boldsymbol{x}_2, ...\boldsymbol{x}_n$ of dimensionality d, the self-attention layer transforms them in the following way.

Firstly, each embedding $\boldsymbol{x}_i$ is projected by three matrices Q, K, and V to get query $\boldsymbol{q}_i$, key $\boldsymbol{k}_i$, and value $\boldsymbol{v}_i$ representations, respectively: $\boldsymbol{q}_i, \boldsymbol{k}_i, \boldsymbol{v}_i \leftarrow Q\boldsymbol{x}_i, K\boldsymbol{x}_i, V\boldsymbol{x}_i$. Next, a scaled dot-product between query $\boldsymbol{q}_i$ and key $\boldsymbol{k}_j$ is calculated as follow:

$$\alpha_{ij} = \frac{1}{\sqrt{d}} \boldsymbol{q}_i \cdot \boldsymbol{k}_j^T \tag{2}$$

This dot-product defines the attention weights $w_{ij} = e^{\alpha_{ij}} / \sum e^{\alpha_{ij}}$ which are used to get the output representations: $\boldsymbol{o}_i = \sum_j w_{ij} \boldsymbol{v}_j$. This process is done in parallel for multiple heads, acting on independent slices of the input embeddings; their outputs are concatenated and passed through a fully connected layer.

3 Transformer Modifications

Self-attention gate (SAG) The simplest way to imitate the effect of GLU activation (Eq. 1) is to weight (gate) the output of self-attention by a learned scalar parameter. To ensure that it is non-negative and is scaled in $[0, 1]$, we parameterize it as a sigmoid of a real-valued learned parameter β. Algorithm 1 illustrates the introduced change. In comparison to Transformer, SAG adds one scalar parameter for each encoder and decoder layer.

We treat β_0, the value β is initialized with before training, as a hyperparameter. In the preliminary experiments, we found that after training, encoder layers often have small negative β values ($-2.. - 0.5$), while decoder layers have positive values ($0.2..4.5$) that grow monotonically for higher layers.

A similar modification was considered in an effort to stabilize Transformer training in the Reinforcement Learning domain (Parisotto et al., 2020).

Convolution as self-attention In the limit case, we can entirely replace self-attention with convolutions. This modification introduces one hyperparameter (kernel size). However, convolutional layers have fewer parameters than the self-attention mechanism. One might consider this not to be a Transformer variant due to the lack of self-attention, but as self-attention generalizes convolutions (Cordonnier et al., 2020), we consider this as an extreme form of regularization.

Fixed-span self-attention A less extreme modification would be to use the regular multi-head self-attention mechanism, but without allowing attention to peek beyond some distance. This mimics the limited kernel size of convolutions in ConvS2S. We achieve this by adding a fixed bias term b_{ij} to the self-attention logits (Eq. 2):

$$\alpha_{ij} = \frac{1}{\sqrt{d}} \boldsymbol{q}_i \cdot \boldsymbol{k}_j^T + b_{ij} \tag{3}$$

Setting b_{ij} to $-\infty$ when the difference $|i - j|$ exceeds some fixed value s and to 0 otherwise prevents the self-attention to look beyond distance s.

Fixed-span self-attention with a span parameter s has the same "receptive field" as 1D convolution with kernel size $2s + 1$. This modification adds one hyperparameter (span size), but does not introduce new learned parameters.

T5 attention Further relaxing constraints on self-attention, we consider the case where we allow Transformer to learn how to (soft-)limit its self-attention. We introduce the bias term b_{ij} that is learned as a function of a (signed) difference $i - j$, capping it to $[-s, +s]$ (e.g., positions with difference above s would have the same bias b_s).

This modification is similar to one introduced by Raffel et al. (2019) in T5, with the only exception that we allow each head to have its own bias. Again, the span size is a new hyperparameter. In a model with n_h heads and n_l layers, this modification requires $(2s + 1) \times n_l \times n_h$ new parameters, which is negligible in comparison with the sizes of fully connected layers. Examples of the learned b_{ij} parameters are shown in Supplementary when training on SCAN.

1: **procedure** COMPUTE_SELFATTENTION
2: $res \leftarrow x$
3: $x \leftarrow self\_attn(x)$
4: $x \leftarrow x * \sigma(\beta)$
5: $x \leftarrow layer\_norm(res + dropout(x))$

Algorithm 1: Self-attention gate (SAG). The only introduced change is on line 4. β is a learned per-layer scalar parameter.

Implementation details We used the fairseq (Ott et al., 2019) implementation of Transformer seq2seq as a foundation, with its initialization and default parameters. T5 and fixed-span attentions are implemented by providing additive attention masks to Transformer.

4 Datasets

4.1 SCAN

Introduced by Lake and Baroni (2018), SCAN is a collection of tasks used for studying systematic generalization of seq2seq models (see Table 1 for some input-output examples). A set of 4 primitive verbs are combined with different modifiers generating around 27k unique samples. Lake and Baroni (2018) and, later, Loula et al. (2018) prepared several non-i.i.d. splits of the data into training and test sets. To successfully generalize on such non-i.i.d. splits, a model has to generalize systematically, in a compositional way.

We experiment with three tasks, often focused on in the literature[2] (Dessì and Baroni, 2019). **SCAN-simple** splits all sequences in train and test sets uniformly at random. Hence, both train and test are identically distributed. Typically models succeed at it easily. In **SCAN-jump**, the test set contains all compositional uses of one of the primitives, `jump`. The train set contains all uses of other primitives, and inputs where `jump` is used in isolation. **SCAN-around-right** tests if a model is capable to generalize to combinations of two modifiers, `around` and `right`, that never co-occur in the training data. The test data contain all examples where the two modifiers are combined.

[2]We have also ensured that our best modification performs on par ($\approx$ 10%) with Transformer on SCAN-length; however SCAN-length is believed to require a different type of generalization (Gordon et al., 2019).

4.2 Machine Translation

We hypothesize that the type of systematic generalization that SCAN probes for could be most useful in data-poor tasks or tasks with train-test distribution shift. Hence, we complement the standard IWSLT'14 En-De dataset with a low-resource task, FLoRes. To study whether our models can perform SCAN-style generalization on natural language data, we also build a dataset that probes for compositional generalization in noun-adjective ordering in French, when translating from English.

IWSLT'14 En-De This is a standard MT dataset, that includes train, validation, & test sets. We apply preprocessing as in the fairseq example.[3]

FLoRes (Guzmán et al., 2019) FloRes is a low-resource dataset for English↔ Nepali and English ↔ Sinhala translation. The dataset is split into dev, devtest, and test subsets. We only use the provided supervised data.

Noun-adjective ordering We take inspiration from SCAN-jump to construct an MT dataset that probes for compositional generalization using noun-adjective ordering in French. In French, both *adjective noun* (forward) and *noun adjective* (backward) orders are used, unlike English that only has the forward order. Which order is used largely depends on the adjective. For example, to refer to a *specific response*, French speakers say *résponse spécifique* (backward order), while *new response* would be *nouvelle résponse* (forward).

To draw a parallel with SCAN-jump, we consider the nouns as primitives and adjectives as modifiers. Modifiers appear with different primitives, however, some primitives appear with only one modifier. For instance, if, in the training set, *response* only appears with *specific* (backward), we test models on translating sentences containing *new response*, where *new* modifies many other nouns in the training set in the forward order. Such generalization is required by humans when dealing with rare or too specific nouns.

To construct our dataset, we start from the English-French Europarl dataset (a part of WMT'14 En-Fr)[4] and select 8 nouns, $\mathcal{N}$ ={*response, institution, organisation, solution, source, decision, responsibility, population*}. We constrain our train set so that each of the nouns in $\mathcal{N}$ appears only with one adjective (hence in

[3]https://github.com/pytorch/fairseq/tree/master/examples/translation

[4]http://www.statmt.org/europarl/

backward order	forward order
(‘specific’, ‘response’)	(‘new’, ‘source’)
(‘particular’, ‘institution’)	(‘good’, ‘decision’)
(‘effective’, ‘organisation’)	(‘big’, ‘responsibility’)
(‘permanent’, ‘solution’)	(‘first’, ‘population’)

Table 2: (adjective, noun) pairs in the train set of the noun-adjective ordering dataset, classified by their order in French language.

one particular order) as shown in Table 2. For example, the noun *response* will only be composed with the adjective *specific*. However, *specific* (and all other adjectives in Table 2) appears with other nouns. To select these sentences, we use the CoreNLP parser (Manning et al., 2014).[5] Finally, all sentences with nouns that are not among the selected ones are kept. In other words, the training set may contain sentences that have neither the selected adjectives nor the selected nouns. This results to 1641681 sentence pairs split into train (1478195 pairs) and validation (163486 pairs) sets. The test set is composed of the filtered sentences of the original Europarl dataset: we select sentences where nouns in the backward column of Table 2 ($\{$*response, institution, organism, solution*$\}$) are only modified by the adjectives in the forward column ($\{$*new, good, big, first*$\}$). Similarly, we also consider the sentences where the nouns of the forward column are composed with the adjectives of the backward column of the Table.[6] This process will ensure that in the test set, the noun-adjective only appears in the reverse order compared to the train set. Unlike the training data, the test data contains only sentences with the target nouns and adjectives. In total, we test models on 659 sentences. Note that the train and validation sets are identically distributed, however, the test set is distribution-shifted w.r.t. train, akin to SCAN-jump.

We follow the preprocessing steps on the fairseq example page for WMT'14 English to French.[3]

5 Methodology

SCAN Lake and Baroni (2018) were concerned by the feasibility of systematic generalization in seq2seq models. Hence, in their experiments, they tuned the models on the train data and then directly evaluated them on test set, reporting test scores. We follow the same protocol: given a grid of hyperparameters, we fit models on the training data. Next, for each hyperparameter configuration, we average the performance of the models across random seeds. Such a setup demonstrates that, at least for some hyperparameter configurations, the introduced models can learn to generalize systematically. At evaluation time, we decode greedily.

IWSLT'14 De-En We run a grid search on train data; next we select the best performing checkpoint on the validation dataset. We report performance on the test data. We use the same training and evaluation protocols as suggested on the fairseq MT example page.[3] We use beam size 5.

FLoRes This dataset has dev, devtest, and test splits provided. We run a hyperparameter grid search training on the dev data. Next, we select the hyperparameter configuration that has the best average (across seeds) performance on devtest. We report the performance of the selected hyperparameter configuration on the test set, averaged across seeds. We use the training/evaluation scripts, tokenization and other parameters suggested on the dataset page: beam size 5 and length penalty 1.2.

Noun-adjective ordering We run the hyperparameter search similarly to IWSLT'14 De-En. The training and evaluation protocols are the ones suggested by the fairseq page for WMT'14 En-Fr.[3] We also use beam size 5.

As we aim to probe abilities for compositional generalization, we introduce an accuracy-based measure, *COMP*. When analyzing models' errors, we encountered 3 common errors: (1) removing the adjective (example 1 in Table 6), (2) replacing the adjective with a synonym and reversing the order (examples 2 and 3 in Table 6), and (3) producing a completely wrong generalization while removing the adjective. While (2) provides a good enough translation, it is a mistake in the noun-adjective order. However, when outputting the right noun and adjective, the order is always preserved. Hence, to measure if a model is compositional, we only look if both the target adjective and the target noun appear in the prediction, irrespective of their order. We define thus COMP as the ratio of predicted sentences that include *both* the target adjective and noun.[7]

[5] https://stanfordnlp.github.io/

[6] We use the Stanford parser to select these sentences.

[7] It happens that models use a synonym in the right order as shown in SAG+T5's prediction 2 in Table 6. In that case, models had generalized well but are still penalized by COMP. COMP is hence only a proxy measure for compositional generalization based on the common failures.

	jump	around-right	simple
Transformer	$3.4_{\pm 2.0}$	$97.6_{\pm 1.5}$	$100.0_{\pm 0.0}$
self-att. gate (SAG)	$17.2_{\pm 5.8}$	$85.2_{\pm 10.0}$	$100.0_{\pm 0.0}$
+ Conv as s.-a.	$25.7_{\pm 20.4}$	$38.4_{\pm 7.8}$	$99.8_{\pm 0.0}$
+ Fixed-span	$33.6_{\pm 9.5}$	$97.6_{\pm 1.3}$	$100.0_{\pm 0.0}$
+ T5	$43.0_{\pm 9.5}$	$92.6_{\pm 2.8}$	$100.0_{\pm 0.0}$
LSTM seq2seq (Lake and Baroni, 2018)	1.2	2.5	99.8
ConvS2S (Dessì and Baroni, 2019)	$69.2_{\pm 8.2}$	$56.7_{\pm 10.2}$	$100.0_{\pm 0.0}$

Table 3: Accuracy on SCAN tasks, %. For each architecture and task, we report the mean accuracy of the best hyperparameter configuration. ± denotes 1 SEM.

Hyperparameter search Transformer models have multiple hyperparameters (embeddings dimensionality, number of layers and attention heads, dropout probabilities, etc.). On top of those, our introduced models add the attention span s, and the initial gate state β_0. For MT tasks, we start from the existing strong baseline hyperparameter configurations (FLoRes: specified by Guzmán et al. (2019), De-En & En-Fr: following the fairseq example page) and only tune (a) the parameters introduced by our architectures, and (b) the attention dropout parameter (for all architectures, including Transformer). For SCAN, there is no baseline hyperparameter configuration, so we start with tuning Transformer and then base hyperparameters of the introduced architectures on it. We report full hyperparameter grids in Supplementary.

6 SCAN experiments

In our preliminary experiments, we found that our modifications of the self-attention mechanism do not lead to improvements over the standard Transformer when they are not combined with the self-attention gate (SAG). Hence, we focus our experiments on architectures that include SAG.

We report our results in Table 3. We also include results for LSTM- and Conv-based seq2seq models that were reported in earlier work (Lake and Baroni, 2018; Dessì and Baroni, 2019). From Table 3, we see that the unmodified Transformer has very low accuracy on `jump` (3.4%), which is only slightly above that of LSTM seq2seq (1.2%) and well below ConvS2S (69.2%). This indicates that Transformer models are indeed failing in compositional generalization on `jump`. However, they have a very high score on the `around-right` split (97.6%) and `simple` ($\geq$ 99.8%). By introducing the different modifications described in Section 3, making Transformers closer to ConvS2S, we aim at preserving the high performance of Transformers on `around-right` and `simple` while significantly improving it on `jump`.

Transformer	SAG	+ Conv s.-a.	+ fix. span	+ T5
$34.64_{\pm 0.03}$	$34.28_{\pm 0.08}$	$33.44_{\pm 0.04}$	$34.32_{\pm 0.01}$	$34.66_{\pm 0.04}$

Table 4: BLEU on test set. IWSLT'14 German to English dataset. ± denotes 1 SEM.

Adding SAG increases accuracy on `jump` 5-fold (17.2%) at the expense of a small drop in `around-right` scores (not stat. sig.).

Further, we observe that changes of the self-attention mechanism (replacing it with Convs, limiting its span, and adding a relative position-dependent bias), can further increase the performance on `jump`. Apart from SAG+Conv as s.-a, the self-attention modifications do not significantly alter the performance on `around-right`.

We see that the architectural changes that we proposed improve the compositional capabilities of the Transformer models. As expected, the introduced hybrid architectures reach significantly better performance on `jump` (up to 12x improvements for SAG+T5) while keeping high performance on the `around-right` & `simple` tasks.

7 Machine Translation experiments

IWSLT'14 De-En In Table 4, we report BLEU scores on German-English translation. SAG + T5 performs slightly better (0.02 BLEU, not stat. sig.), but other modifications underperform w.r.t. Transformer. Replacing self-attention with convolutions resulted in the largest drop, 3%. Other differences are smaller, $\leq$ 1%. For the following parts, we only experiment with the SAG + T5 model as the only non-degraded one. However, results with the remaining models on FLoRes and the Noun-adjective ordering datasets are reported in Supplementary.

FLoRes, Si/Ne $\leftrightarrow$ En We report results on

English↔Nepali and English↔Sinhala translation in Table 5. Following Guzmán et al. (2019), we use tokenized BLEU when translating from English. We run standard Transformer models as specified in Guzmán et al. (2019), but adding a search over the attention dropout probability. We verify that we have close results compared to Guzmán et al. (2019).[8] Table 5 shows that SAG + T5 outperforms Transformer on all language pairs and directions with relative improvements between 3.6% (si-en) and 13.1% (en-ne).

Noun-adjective ordering BLEU scores on the test set are reported in Table 5. SAG + T5 leads to a relative improvement of 1.39% compared to standard Transformer. BLEU, however, is not informative about the particular noun-adjective generalization. We hence also report COMP scores. From Table 5, we see that SAG + T5 demonstrates a significant improvement with 14% relative gain compared to the standard Transformer architecture. Our follow-up experiments show that the hybrid model recovers an average of 43.3% of cases where the best Transformer model (best seed w.r.t. COMP) failed in compositional generalization, whereas Transformer is only correct at 21.5% of SAG + T5's errors. We report in Table 6 examples comparing SAG + T5 and Transformer translations.

Discussion Upon analyzing our experiments on SCAN and machine translation tasks, we see the following picture. Indeed the hybrid models that we described in Section 2 have considerably higher accuracy on SCAN-`jump` w.r.t. Transformer and a comparable performance on the other SCAN splits. Hence, our results suggest the importance of both gating and (the ability of) limiting the attention span for SCAN generalization.

As expected, the improvement on SCAN do not consistently entail improvements on the resource-rich dataset, and only the combination of SAG and T5 showed a tiny improvement. This emphasizes the importance of testing models on realistic setups to model from being too SCAN-tailored.

Finally, we test SAG + T5 on low-resource and compositional tasks. The hybrid architecture shows consistent improvements on FLoRes for all translation directions, with at up to 13.1% relative improvement, and on the the natural language compositional task with 14% relative improvement on COMP. Our qualitative analysis also showed that SAG + T5 correctly handles noun-adjective ordering in most cases, while Transformer makes more mistakes.

[8] We got better BLEU scores due to the extra grid search.

8 Related Work

Compositionally-biased models Several approaches were proposed to build SCAN-capable architectures. They span from meta-learning (Lake, 2019), disentangling syntax and semantics (Russin et al., 2019), learning equivariant (Gordon et al., 2019) and disentangled representations (Li et al., 2019) or combining neural & symbolic computations (Chen et al., 2020). In contrast, we do not build new models that are specialized to SCAN. Instead, we show that a standard model can be incrementally modified so that performs well on SCAN and still performs well on a standard MT task. Having such incrementally improved models allows us to step back and wonder if SCAN (or similar artificial tasks) should be used as a guidance when developing new models.

Bastings et al. (2018) raised concerns due to SCAN being too artificial by showing that even degenerate architectures can perform well on some SCAN tasks. Our results echo their findings: by developing architectures tailored for SCAN, one can easily come up with models that perform worse on general tasks. However, we find that if one avoids this "SCAN overfitting" and endows a model with capabilities that SCAN probes for without harming its general performance, they can gain in low-resource scenarios and better handle relevant phenomena in language.

Changing attention mechanisms Self- and cross-attention mechanisms were tweaked in earlier work in order to inject useful biases, e.g., by adding information of relative positions of tokens (Shaw et al., 2018; Raffel et al., 2019) or accounting for the locality bias in cross-attention (Yang et al., 2018). Sukhbaatar et al. and Rae and Razavi (2020) demonstrated that having a short attention span on the lower layers of Transformer models is enough for good language modeling performance.

9 Conclusion

In this work, we primarily focused on whether and in which scenarios the inductive bias for compositional generalization, that SCAN looks for, can be useful in natural language tasks.

We ran study in two steps. As the first step, by exploiting ConvS2S/Transformer similarities, we came up with a modification of the Transformer

	FLoRes (BLEU)				Noun-Adj. ordering	
	ne-en	si-en	en-ne	en-si	BLEU	COMP
Transformer	$7.94_{\pm 0.05}$	$7.15_{\pm 0.07}$	$4.43_{\pm 0.01}$	$2.32_{\pm 0.08}$	$40.86_{\pm 0.34}$	$0.64_{\pm 0.01}$
SAG + T5	$\mathbf{8.40_{\pm 0.02}}$	$7.41_{\pm 0.10}$	$\mathbf{5.01_{\pm 0.10}}$	$\mathbf{2.54_{\pm 0.03}}$	$41.43_{\pm 0.29}$	$\mathbf{0.73_{\pm 0.01}}$

Table 5: Models performance on FLoRes and Noun-adjective ordering (English to French) dataset. For FLoRes, we report the BLEU dev-test scores for the different translation directions. For the Noun-adjective ordering dataset, we report both BLEU and COMP measures on the test set. In bold are values that stat. sig. improve over Transformer. $\pm$ denotes 1 SEM.

Target: Nous sommes face à une *responsabilité politique particulière*.
Prediction SAG+T5: Nous sommes accablés par une *responsabilité politique particulière*.
Prediction Transformer: Nous sommes accablés par une *responsabilité politique*.
Target: Nous voulons trouver une *bonne solution* à ce problème.
Prediction SAG+T5: Nous voulons trouver une *bonne solution* à ce problème.
Prediction Transformer: Nous voulons trouver une *solution adéquate* à ce problème.
Target: Ce qui nous déçoit par rapport à cette *décision particulière*, c'est que le projet aurait pu clairement voir le jour.
Prediction SAG+T5: Ce qui est triste dans cette *décision précise*, c'est que le projet aurait été clairement réalisé.
Prediction Transformer: Ce qui est triste dans cette *mauvaise décision*, c'est que le projet aurait clairement été.

Table 6: Generation Examples for Noun-adjective ordering dataset. Models are tested on the underlined and italic *(adjective, noun)*. For the first 2 examples, SAG+T5 predicted the right *(adjective, noun)* translation. In the last one, SAG+T5 replaced the adjective with a synonym but in the right target order (the one not seen in the training set). In the first example, Transformer removed the adjective *particulière*. In the two following examples, Transformer replaced the right adjective with a close synonym adjective to be conform with the training order. For instance, in the second example, *bonne* (an adjective that appears in the forward order) was replaced by *adéquate* (an adjective that appears in the backward order) as the *solution* appears only in the backward order at training.

architecture that performs considerably better than vanilla Transformer on SCAN-jump (43.0% vs 3.4% accuracy) and performs equally well on SCAN-simple, SCAN-around-right, and on a standard resource-rich MT task (IWSLT'14 De-En).

Next, we tested this modification in low-resource and distribution-shifted setups. In the low-resource MT setup (FLoRes Si/Ne↔En), we found that our considered architecture improves by up to 13.1% in BLEU score over the vanilla Transformer. Then, we introduced a new dataset that probes specifically for compositional reasoning in natural language. Unlike SCAN, our compositional dataset is built by filtering an existing natural language corpus (EuroParl En-Fr) to probe how models perform noun-adjective ordering under a (minimal) distribution shift. Thus, we are largely closer to testing the compositional generalization required by humans compared to SCAN, and succeeding on the test set requires both compositional reasoning and good language model performance (see examples in Table 6). We believe that such a dataset is beneficial for future research to test more complex compositionality skills. Finally, our experiments on our dataset demonstrated that better SCAN generalization leads to better results on noun-adjective ordering (14% on COMP).

Our findings indicate the following. Firstly, as hypothesized before (Dessì and Baroni, 2019; Hupkes et al., 2018), the limited attention span provides a useful inductive bias that allows models to perform better on compositional generalization induction, that SCAN probes for. Further, endowing a model with SCAN-style generalization capabilities *can* lead to improvements in low-resource and distribution-shifted scenarios as long as we ensure that we do not overfit to SCAN.

We believe that the contribution of diagnostic datasets like SCAN is of great value. As performance grows on tasks such as MT, identifying gaps where a model's performance lags will become fundamental and will guide us to develop architectures that cover genuine new linguistic grounds and not just overfit to peculiarities of standard datasets.

Acknowledgments

The authors are grateful to Marco Baroni and the reviewers for feedback that helped us to improve our .

References

Daniel Adiwardana, Minh-Thang Luong, David R So, Jamie Hall, Noah Fiedel, Romal Thoppilan, Zi Yang, Apoorv Kulshreshtha, Gaurav Nemade, Yifeng Lu, et al. 2020. Towards a human-like open-domain chatbot. *arXiv preprint arXiv:2001.09977*.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Marco Baroni. 2019. Linguistic generalization and compositionality in modern artificial neural networks. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 375(1791):20190307.

Jasmijn Bastings, Marco Baroni, Jason Weston, Kyunghyun Cho, and Douwe Kiela. 2018. Jump to better conclusions: Scan both left and right. In *MNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*.

Xinyun Chen, Chen Liang, Adams Wei Yu, Dawn Song, and Denny Zhou. 2020. Compositional generalization via neural-symbolic stack machines.

Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. 2020. On the relationship between self-attention and convolutional layers. In *ICLR*.

Verna Dankers, Elia Bruni, and Dieuwke Hupkes. 2021. The paradox of the compositionality of natural language: a neural machine translation case study.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *ICML*.

Roberto Dessì and Marco Baroni. 2019. CNNs found to jump around more skillfully than RNNs: Compositional generalization in seq2seq convolutional networks. In *ACL*.

Richard Futrell, Kyle Mahowald, and Edward Gibson. 2015. Large-scale evidence of dependency length minimization in 37 languages. *PNAS*, 112(33):10336–10341.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In *ICML*.

Jonathan Gordon, David Lopez-Paz, Marco Baroni, and Diane Bouchacourt. 2019. Permutation equivariant models for compositional generalization in language. In *ICLR*.

Francisco Guzmán, Peng-Jen Chen, Myle Ott, Juan Pino, Guillaume Lample, Philipp Koehn, Vishrav Chaudhary, and Marc'Aurelio Ranzato. 2019. Two new evaluation datasets for low-resource machine translation: Nepali-english and sinhala-english.

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

Dieuwke Hupkes, Anand Singh, Kris Korrel, German Kruszewski, and Elia Bruni. 2018. Learning compositionally through attentive guidance. *arXiv preprint arXiv:1805.09657*.

Eugene Kharitonov and Rahma Chaabouni. 2021. What they do when in doubt: a study of inductive biases in seq2seq learners. In *ICLR*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Brenden Lake. 2019. Compositional generalization through meta sequence-to-sequence learning. In *NeurIPS*.

Brenden Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *ICML*.

Yuanpeng Li, Liang Zhao, Jianyu Wang, and Joel Hestness. 2019. Compositional generalization for primitive substitutions. *arXiv preprint arXiv:1910.02612*.

Joao Loula, Marco Baroni, and Brenden M Lake. 2018. Rearranging the familiar: Testing compositional generalization in recurrent networks. *EMNLP 2018*.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *ACL: System Demonstrations*.

R Thomas McCoy, Robert Frank, and Tal Linzen. 2020. Does syntax need to grow on trees? sources of hierarchical inductive bias in sequence-to-sequence networks. *TACL*, 8:125–140.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *NAACL-HLT 2019: Demonstrations*.

Emilio Parisotto, Francis Song, Jack Rae, Razvan Pascanu, Caglar Gulcehre, Siddhant Jayakumar, Max Jaderberg, Raphael Lopez Kaufman, Aidan Clark, Seb Noury, et al. 2020. Stabilizing transformers for reinforcement learning. In *ICLR*.

Jack Rae and Ali Razavi. 2020. Do transformers need deep long-range memory? In *ACL*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Jake Russin, Jason Jo, Randall C O'Reilly, and Yoshua Bengio. 2019. Compositional generalization in a deep seq2seq model by separating syntax and semantics. *arXiv preprint arXiv:1904.09708*.

Tim Salimans and Durk P Kingma. 2016. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *NeurIPS*.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *NAACL*, pages 464–468.

Sainbayar Sukhbaatar, Édouard Grave, Piotr Bojanowski, and Armand Joulin. Adaptive attention span in transformers.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. 2013. On the importance of initialization and momentum in deep learning. In *ICML*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *NeurIPS*.

Baosong Yang, Zhaopeng Tu, Derek F Wong, Fandong Meng, Lidia S Chao, and Tong Zhang. 2018. Modeling localness for self-attention networks. *arXiv preprint arXiv:1810.10182*.

A Hyperparameter grids

A.1 SCAN

For each architecture, we used the same hyperparameter grids for all splits of SCAN.

All models were trained by Adam with default β_1 and β_2 parameters, for 250 epochs, with batch size 256, learning rate $5 \cdot 10^{-4}$, dropout and attention dropout 0.1, random seeds $\{0, 1, 2\}$. We vary the Encoder and Decoder parameters independently: number of attention heads $\{4, 8\}$, embedding dimensions $\{128, 256\}$, FFN dimensions $\{256, 512\}$, and the number of layers $\{4, 6, 8\}$; clip norm 1.0.

For hybrids models, add the following parameters.

SAG To reduce the search space, we did not vary β_0, setting it to -1.

SAG + CNN as self-attention Span: $\{2, 4, 6\}$, $\beta_0 = -1$, number of layers: $\{4, 6\}$.

SAG + fixed span self-attention Span: $\{2, 4, 6\}$, $\beta_0 = -1$, number of layers: 4.

SAG + T5 Span: $\{2, 4, 6\}$, $\beta_0 = -1$, number of layers: 4.

A.2 Machine Translation

De-En We start from the standard architecture suggested by Fairseq examples for IWSLT'14 De-En. That is we share decoder input and output embeddings. Both Encoder and decoder have an embedding size of 512 FFN dimensions of 1024, 4 attention heads, 6 encoder & decoder layers. We used adam optimizer with learning rate of 5e-4, no clip norm, warm-up learning rate 1e-7, inverse square root learning rate scheduler, 4000 warm-up updates, dropout 0.3, weight decay 1e-4, label smoothing 0.1, max. tokens per batch per GPU: 1024, 40 epochs. We used 4 GPUs for training.

For SAG-enabled architectures, we additionally searched for Encoder's β_0 in $\{-1, 0\}$ and $\{1, 0\}$ for Decoder. We varied attention span in $\{2, 4, 6\}$, ttention dropout in $\{0.0, 0.2\}$ and pre-block encoder and decoder normalization.

For model selection, we also follow Fairseq example and checkpoint the best model on validation set based on BLEU score. BLEU score is computed with a beam of size 5.

FLoRes We used shared embeddings between Encoder and Decoder, embedding dimenions of 512, FFN dimensions of 2048, 2 attention heads, 5 encoder & decoder layers. We used pre-block normalization[9], learning rate of 1e-3, no clip norm, warm-up learning rate 1e-7, inverse square root learning rate scheduler, 4000 warm-up updates, dropout 0.4, activation dropout 0.2, weight decay 1e-4, label smoothing 0.2, max. tokens per batch per GPU: 4000, 100 epochs. We searched for attention dropout in $\{0.0, 0.2\}$. We used 4 GPUs for training.

For SAG-enabled arcchitectures, we additionally searched for Encoder's β_0 in $\{-2, -1, 0\}$ and $\{2, 1, 0\}$ for Decoder. We varied attention span in $\{2, 4, 6\}$.

Noun-adjective order agreement We start from the standard architecture suggested by Fairseq examples for WMT'14 En-Fr. That is we share encoder, decoder and output embeddings. Both Encoder and decoder have an embedding size of 1024 FFN dimensions of 4096, 16 attention heads, 6 encoder & decoder layers. We used adam optimizer with learning rate of 7e-4, no clip norm, warm-up learning rate 1e-7, inverse square root learning rate scheduler, 4000 warm-up updates, dropout 0.1, label smoothing 0.1, max. tokens per batch per GPU: 4000, 30 epochs. We used 6 GPUs for training.

For SAG-enabled architectures, we additionally searched for Encoder's β_0 in $\{-1, 0\}$ and $\{1, 0\}$ for Decoder. We varied attention span in $\{2, 4, 6\}$, ttention dropout in $\{0.0, 0.2\}$ and pre-block encoder and decoder normalization.

Best checkpoint is based on the loss of the validation set.

B Other modifications on FloRes and Noun-adjective ordering datasets

In the main paper, we only experimented with SAG + T5 as the only non-degraded modification on the IWSLT'14 En-De dataset. Our intuition is that the remaining hybrid models are SCAN-tailored and would not lead to any improvement in the low-resource (FloRes) and domain-shifted (Noun-adjective ordering dataset) settings. In this section, we verify our intuition and report the results of all the introduced variants. The hyper-parameters search is reported in Section A.

FloRes We report results on English↔Nepali and English↔Sinhala translation in Table 7. We

[9] `--encoder-normalize-before` and `--decoder-normalize-before` in fairseq.

also report Guzmán et al. (2019) results under "Baseline".

Analyzing results of SAG, we notice that it is usually very close to Transformer's results on all tasks, apart from Nepali→English, where it lags behind. The fixed-span modification performs worse than Transformer on in all directions. Replacing self-attention with convolutions results in better scores on En→Ne and worse scores on Ne→En/En→Si.

Hence, as expected, only the SAG + T5 model outperforms Transformer on all language pairs and directions, highlighting the importance of verifying the generality of the model on realistic datasets.

Noun-adjective order agreement BLEU scores on the test set are reported in Table 8. SAG leads to a relative improvement of 1.44% compared to standard Transformer, closely followed by SAG + T5. Still, in total, the differences are very small across all models. On the other hand, all introduced variants outperform standard Transformer on COMP. However, only SAG + T5 demonstrates a significant improvement with 14% relative gain.

Overall, we observe that the SCAN-tailored variants do not degrade performances on the Noun-adjective order agreement dataset, but still do not lead to any significant improvement, contrary to SAG + T5.

C Visualizing attention biases

In this Section, we illustrate how a successful SAG + T5 model uses its b_{ij} terms (Eq. 1) to control its attention.

We take the most successful hyperparameter combination on SCAN-`jump` in Table 3 and select a model instance that has the best accuracy ($\approx 60\%$). Next, for each attention head of each encoder and decoder layer, we retrieve its learned relative-position bias b_d, where d is a (signed) relative distance between positions i and j, that is capped to be within $[-s, +s]$ (see Section 3). For each head, we apply a softmax, to find its "preference" $\hat{b}_d$ over relative positions d:

$$\hat{b}_d = \frac{\exp(b_d)}{\sum_d \exp(b_d)}$$

We report the results in Figure 1. Interestingly, quite a few attention heads have very strong preferences for fixed relative positions and some are even dominantly focused on particular positions (Encoder: head 7 in the layer 0; heads 4, 5 in layer 1, heads 3, 7 in layer 2; head 2 in layer 3; Decoder: head 4 in layer 0, head 2 in layer 1, heads 3,4,5 in layer 2; heads 2, 6, 7 in layer 3)[10]. More often than not, those "specialized" heads look within the span and not on the "border" values of d (due to d being capped, they also correspond to arbitrary distant positions to the left and right).

Hence we conclude that in a T5 model (among most successful on SCAN-`jump`), several heads leverage the ability pay attention locally; supporting our finding that local attention is connected with the compositional generalization needed to succeed at SCAN. At the same time, some heads have large relative-position bias for distant positions ($[s, +\infty[$ or $]-\infty, -s]$). This general ability to *optionally* look beyond a fixed span in T5 could be responsible for its better performance compared to the fixed span modification.

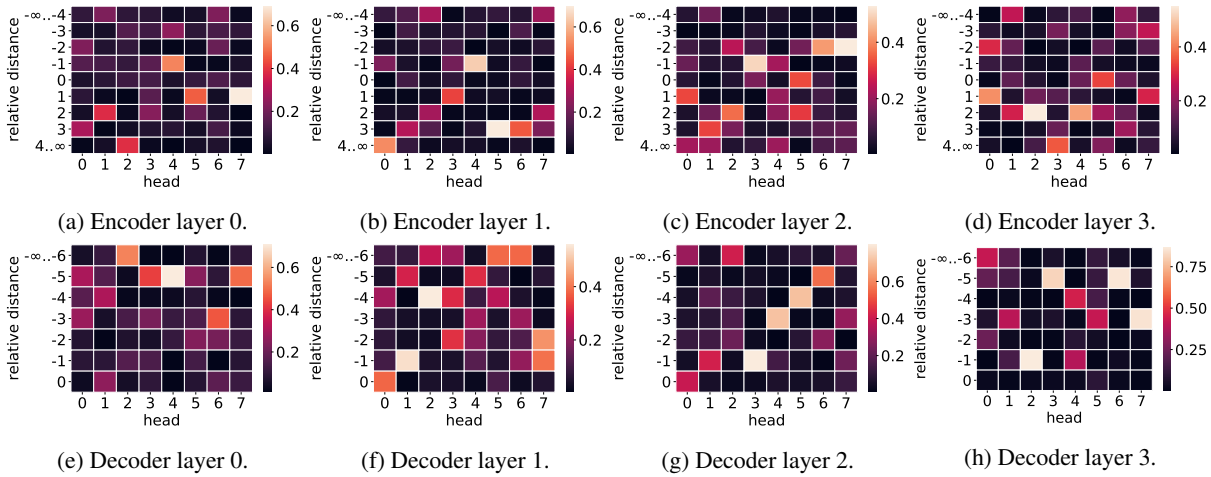
[10] T5 reduces to the vanilla Transformer if all b_d are equal to zero. That corresponds to the uniform bias $\hat{b}_d$.

	Baseline	Transformer	SAG	+ Conv s.-a.	+ fixed span	+ T5
ne-en	7.6	$7.94_{\pm 0.05}$	$7.58_{\pm 0.06}$	$7.59_{\pm 0.02}$	$7.44_{\pm 0.08}$	$\mathbf{8.40_{\pm 0.02}}$
si-en	7.2	$7.15_{\pm 0.07}$	$7.14_{\pm 0.10}$	$7.18_{\pm 0.10}$	$.78_{\pm 0.07}$	$7.41_{\pm 0.10}$
en-ne	4.3	$4.43_{\pm 0.01}$	$4.36_{\pm 0.08}$	$\mathbf{4.63_{\pm 0.03}}$	$4.12_{\pm 0.05}$	$\mathbf{5.01_{\pm 0.10}}$
en-si	1.2	$2.32_{\pm 0.08}$	$2.37_{\pm 0.10}$	$2.14_{\pm 0.03}$	$2.12_{\pm 0.04}$	$\mathbf{2.54_{\pm 0.03}}$

Table 7: BLEU dev-test scores on FLoRes. Baseline scores are taken from (Guzmán et al., 2019). In bold are values that stat. sig. improve over Transformer ($p < 10^{-3}$). $\pm$ indicates 1 SEM.

	Transformer	SAG	+ Conv s.-a.	+ fixed span	+ T5
BLEU	$40.86_{\pm 0.34}$	$41.45_{\pm 0.14}$	$39.89_{\pm 0.27}$	$41.01_{\pm 0.24}$	$41.43_{\pm 0.29}$
COMP	$0.64_{\pm 0.01}$	$0.70_{\pm 0.03}$	$0.67_{\pm 0.01}$	$0.68_{\pm 0.01}$	$\mathbf{0.73_{\pm 0.01}}$

Table 8: BLEU and COMP measures on test sets: compositional English to French dataset. In bold are values that stat. sig. improve over Transformer ($p < 0.05$). $\pm$ denotes 1 SEM.



(a) Encoder layer 0. (b) Encoder layer 1. (c) Encoder layer 2. (d) Encoder layer 3.

(e) Decoder layer 0. (f) Decoder layer 1. (g) Decoder layer 2. (h) Decoder layer 3.

Figure 1: Relative attention biases for T5 + SAG architecture (after a softmax). Each cell indicates preference of a head to a position at a signed relative distance. The relative distances are capped. For the decoder we only represent relative attention biases for $d \leq 0$, as positions with positive relative distance are masked in the autoregressive decoder.