Monotonic Location Attention for Length Generalization

Jishnu Ray Chowdhury [1]   **Cornelia Caragea** [1]

Abstract

We explore different ways to utilize position-based cross-attention in seq2seq networks to enable length generalization in algorithmic tasks. We show that a simple approach of interpolating the original and reversed encoded representations combined with relative attention allows near-perfect length generalization for both forward and reverse lookup tasks or copy tasks that had been generally hard to tackle. We also devise harder diagnostic tasks where the relative distance of the ideal attention position varies with timestep. In such settings, the simple interpolation trick with relative attention is not sufficient. We introduce novel variants of location attention building on top of Dubois et al. (2020) to address the new diagnostic tasks. We also show the benefits of our approaches for length generalization in SCAN (Lake & Baroni, 2018) and CFQ (Keysers et al., 2020). Our code is available on GitHub[1].

1. Introduction

Neural seq2seq (Sutskever et al., 2014) is a powerful generic framework for the task of transforming an input sequence of arbitrary length into an output sequence of arbitrary length. Although seq2seq models can perform impressively in a great variety of tasks (Raffel et al., 2020; Lewis et al., 2020), they can still struggle in out-of-distribution generalization (e.g., systematic generalization or length generalization), and sometimes, even in simple algorithmic tasks (Kim et al., 2022; Dubois et al., 2020; Dehghani et al., 2019; Lake & Baroni, 2018; Liska et al., 2018). Even after extensive pre-training, neural models can show mixed results in such forms of generalization (Kim et al., 2022; Anil et al., 2022).

[1]Computer Science, University of Illinois Chicago. Correspondence to: Jishnu Ray Chowdhury <jraych2@uic.edu>, Cornelia Caragea <cornelia@uic.edu>.

[1]https://github.com/JRC1995/MonotonicLocationAttention

In this paper, we focus on length generalization, i.e., the ability of a model to generalize to sequences of unseen (and typically higher) lengths. Particularly, we concentrate on enhancing the interlayer attention mechanism in seq2seq encoder-decoder models for improved length generalization. Similar to Csordás et al. (2022), we take a *bottom up* approach to model development and explore the effects of different strategies of increasing complexities on a range of controlled synthetic probing tasks—each targeting a narrowly defined model behavior or phenomenon—to investigate which strategy works and to what extent, and why does or does not work, and thus, each task precisely pinpointing their capabilities as well as their limitations. Such thorough investigation in a natural language domain can be difficult for at least the following reasons: (1) it can be hard to isolate the exact reasons of failure in natural language due to its complexities and diversity; (2) often there can be exploitable heuristics like emphasis on recency that may improve the overall length generalization performance but preserve systematic issues leading to failures in cases where the heuristics do not apply. Such failures may not be reflected in the overall evaluation if the heuristics apply in the majority of the samples. Besides these factors, the simple synthetic tests that we consider here can be still fairly challenging for neural models. We believe they offer an initial step toward the design of more general-purpose models.

To achieve the above desideratum and evaluate length generalization capability of different interlayer attention mechanisms, we set up ten synthetic probing task (see Table 1 and §2). Following prior work (Graves et al., 2014; Dehghani et al., 2019; Liang et al., 2021), we first consider the task of simply copying source texts in both forward and backward (reverse) directions. Following Dubois et al. (2020), we also consider compositional lookup table task (Liska et al., 2018) in both directions. However, as we will show in §2, in these tasks the ideal attention position can be trivially determined from the decoding timestep alone—a condition (let us call it C1) that simply allows the relative positional attention (Shaw et al., 2018; Dai et al., 2019) to perform perfectly given the right implementation. Thus, we propose new probing tasks involving repeated copy (ReCopy) and its variants to create settings where C1 is not satisfied. While there are already more synthetic tasks where C1 is not satisfied, our proposed tasks (ReCopy and its variants) are intended to be

Task	Input	Output
Copy	4 7 9 8	4 7 9 8
Reverse Copy	4 7 9 8	8 9 7 4
Lookup	010 $t3$ $t4$ $t2$ $t6$ $t1$.	010 011 010 011 001 001
Reverse Lookup	$t1$ $t6$ $t2$ $t4$ $t3$ 010 .	010 011 010 011 001 001
ReCopy	4 7 9 8	4 4 4 7 7 7 7 7 9 9 9 9 8 8 8 8 8
Reverse ReCopy	4 7 9 8	8 8 8 8 8 9 9 9 9 9 7 7 7 7 7 4 4 4
Inv ReCopy	4 4 4 7 7 7 7 7 9 9 9 9 8 8 8 8 8	4 7 9 8
Inv Reverse ReCopy	8 8 8 8 8 9 9 9 9 9 7 7 7 7 7 4 4 4	4 7 9 8
SCAN	look and run right	I_LOOK I_TURN_RIGHT I_RUN

Table 1. Input-output examples for each task (except CFQ).

small extensions over simple copy tasks such that the exact cause of model failure can be clearer compared to more complex tasks. Not only do we propose new probing tasks, but we also propose new strategies to tackle them. Prior models (Dehghani et al., 2019; Dubois et al., 2020) already struggled in reverse copy or reverse lookup tasks. We introduce a technique of interpolating forward and reversed encoded representations to handle reverse direction even with simple relative attention (the technique is universally applicable to any seq2seq architecture). Moreover, we also propose new attention models, OneStep attention and monotonic location attention (our full model), to handle the proposed probing tasks on which the prior models fail. We also show that our models maintain comparable performance in the SCAN (Lake & Baroni, 2018) (a dataset for translating simple commands into sequences of actions) and CFQ (Keysers et al., 2020) length splits (a dataset for query-to-SQL translation).

2. Probing Tasks

We now describe the ten[2] probing tasks used in this paper. We present examples for each task in Table 1.

Copy: The copy task requires copying input tokens into the output tokens. In this case, the encoder-decoder network has to simply learn an identity function ($x = f(x)$). For this task we use a vocabulary of natural number tokens from 0-9 (see Table 1 for an example). We generated $10,000$ training samples with sequence length in the range 5-10. For the development set, we generated $2,000$ samples of sequence length 10-15. For test sets, we generated a split with sequence length 15, another split with sequence length 30, and another with sequence length 100. Each test split has $2,000$ samples.

Reverse Copy: In the reverse copy task, the model has to copy the input as above but in the reverse direction (see Table 1 for an example). This task is generated with the same parameters as the copy task.

Lookup: Lookup represents the "Long Lookup Tables"

task (Liska et al., 2018) as made available in the code.[3] For any input like "001 $t1$ $t2$ $t3$.", the output for this task will be "$v1$ $v2$ $v3$ $v4$" where $v1 = 001$, $v2 = t1(001)$, $v3 = t2(t1(001))$, and $v4 = t3(t2(t1(001)))$. Here, $t1$, $t2$, and $t3$ are functions, each corresponding to some lookup table such that $t_i : \{0,1\}^3 \rightarrow \{0,1\}^3$ (for any natural number i). The task is generated using the supplied code.[3] The code generates a training split of approximately $9,000$ samples of lengths ≤ 6. We consider three generated test splits that are of sequence lengths 7, 9, and 11. The first test split has approximately $4,500$ samples whereas the others have approximately $5,000$ samples. The development split consists of about 500 samples of sequence length ≤ 6 and approximately 500 samples of length 7.

Reverse Lookup: Reverse Lookup represents the "Long Lookup Tables Reverse" task (Liska et al., 2018) as can be generated from the code.[3] For any input like "$t1$ $t2$ $t3$ 001 .", the output for this task will be "$v1$ $v2$ $v3$ $v4$" where $v4 = 001$, $v3 = t3(001)$, $v2 = t2(t3(001))$, and $v1 = t1(t2(t3(001)))$. Here, $t1$, $t2$, and $t3$ are lookup functions as before. The splits of this task are created similarly to those of the Lookup task described above.

ReCopy: There is one thing that is common in the above tasks. For the forward tasks (Lookup, Copy), assuming that the encoder can keep the information of position i at position i after encoding with necessary contextualization (e.g., composition of previous functions in case of Lookup), the ideal encoding position to attend during decoding will always remain at the same constant distance from the decoding timestep. This is also true for the reversed tasks (Reverse Copy, Reverse Lookup) if the encoding is reversed. For example, to copy "4 7 9 8", at timestep 1 the model has to attend position 1 to print 4. Next, at timestep 2 the model has to attend position 2 to print 7. Thus, more generally, for any timestep t the model has to attend an encoding position i such that $i - t = c$ (where c is some constant. In this example, $c = 0$). Even more generally, in all these tasks, the ideal position to attend can be determined just based on

---

[2]Twelve including tasks in Appendix A.

the decoding timestep t. For instance, for the above tasks, the ideal position of attention i can be defined as a function over timestep as $i = f(t) = t + c$. However, such a happy situation will not be maintained in more complex tasks.

Thus, we decide to create a new set of diagnostic/probing tasks that are close to the previous tasks but precludes the possibility of determining the position of attention just from the timestep. With this motivation, first, we introduce the task ReCopy (Repeated Copy). In this task, the vocabulary includes natural numbers in the range 0-9. If the input for the task is "4 7 9 8", then the corresponding output will be "4 4 4 7 7 7 7 7 9 9 9 9 9 8 8 8 8 8". Effectively, in this task, the model has to learn to not just copy but also to repeat the copied item for a certain frequency before copying the next item. There is a specific set of rules behind how many times an item should be repeated. That is, if the item is a number ≤ 3 the model should print it once, if the item is a number x such that $3 < x \leq 6$ the model should print it three times, and for any other number > 6, the model should print it five times. The splits and sample sizes for this task are similar to those of the copy task.

Our intention here is to make a small extension of the copy task that avoids determination of the attention position purely from the timestep but without introducing any additional sources of difficulty so that the causes of failures can be disentangled more easily. For instance, if a model succeeds in the copy task but fail in ReCopy we can now reasonably infer that its cause of failure is the specific difficulty introduced in ReCopy. Note that if we made ReCopy a bit simpler by requiring each number to be copied and repeated for a uniform frequency, then the determination of the ideal position for attention will again become trivially possible just from a decoding timestep; thus ReCopy requires repeating with varying frequency depending on which number is being copied.

Reverse ReCopy: The Reverse ReCopy task is similar to the ReCopy task in all aspects other than the fact that the copying takes place in the reversed direction (see example in Table 1). The task splits are generated in the same way as in the Copy task.

Inv ReCopy: The Inv ReCopy task (Inverse ReCopy) is similar to the ReCopy task in all aspects other than the fact that the inputs and outputs are inverted (see example in Table 1). The task splits are generated in the same way as in the Copy task.

Inv Reverse ReCopy: The Inv Reverse ReCopy task (Inverse Reverse ReCopy) is similar to the Reverse ReCopy task in all aspects other than the fact that the inputs and outputs are inverted (see example in Table 1). The task splits are generated in the same way as in the Copy task.

SCAN: SCAN is a popular dataset used for testing system-atic generalization (Lake & Baroni, 2018). It involves the task for translating simple commands into a sequence of actions. We explore its original length generalization split.

CFQ: CFQ is a realistic semantic parsing dataset (Keysers et al., 2020) proposed for evaluating compositional generalization. We explore its length generalization split.

We also propose and explore two additional probing tasks (**DeDupe** and **PosRetrieve**) in Appendix A.

3. Seq2Seq General Framework

A seq2seq model can be formalized as a function $F_{seq2seq} :$ $\mathbb{N}^s \rightarrow \mathbb{N}^z$ that maps some input sequence $x_{1:s} = (x_1, x_2, \ldots, x_s)$ of length s to an output sequence $y_{1:z} = (y_1, y_2, \ldots, y_z)$ of length z. Here each element in $x_{1:s}$ and $y_{1:z}$ is a natural number that indexes some distinct token from a vocabulary. $F_{seq2seq}$ constitutes two major components: an encoder function (F_{enc}) and a decoder function (F_{dec}). The encoder $F_{enc} : \mathbb{N}^s \rightarrow \mathbb{R}^{s \times d}$ maps the initial token indices $x_{1:s}$ to a sequence of hidden state representations $e_{1:s} = (e_1, e_2, \ldots, e_s)$ (where any $e_i \in \mathbb{R}^d$). The decoder $F_{dec} : \mathbb{N}^* \times \mathbb{N} \rightarrow \mathbb{N}$ generates the output sequence $y_{1:z}$ recursively one token at a time, typically in an autoregressive manner. That is, at each timestep t (beginning at $t = 1$), F_{dec} takes as input the history of all previously generated tokens $H^{t-1} = (go, y_1, y_2, \ldots, y_{t-1})$ and the last generated token index y_t and outputs y_{t+1}. H^0 is initialized with (go) where go represents the index of a special token that marks the beginning of the generation.

One salient component within the decoder is an interlayer (cross) attention function (Bahdanau et al., 2015) that allows the decoder to interact and retrieve encoded state information. The decoder, in timestep t, will typically create history-contextualized representation $h^{t-1} \in \mathbb{R}^d$ (compressing H^{t-1}). Let query $q_{t-1} = f_q(h_{t-1})$, keys $k_i = f_k(e_i)$, and values $v_i = f_v(e_i)$ ($\forall i \in \{1, \ldots, s\}$) where f_q, f_k, and f_v are linear transformations ($f_{q|k|v} : \mathbb{R}^d \rightarrow \mathbb{R}^d$). In the attention layer, the query interacts with the keys to score each corresponding value. A weighted sum of values based on those scores is then computed as the result of the attention function. This allows the decoder to dynamically and softly retrieve information from any position in the encoded representations $e_{1:s}$ at any timestep. For our work, we explore a variety of cross-attention functions which we discuss below.

4. Prior Approaches to Cross-Attention

4.1. Content Attention

As a baseline, we use the popular scaled inner dot-product query-key based attention as used in Vaswani et al. (2017):

$$c_{ti} = \frac{< q_t, k_i >}{\sqrt{d}}, \tag{1}$$

$$a_{ti} = \frac{\exp(c_{ti})}{\sum_{j=1}^{s} \exp(c_{tj})}, \; o_t = f_o(\sum_{j=1}^{s} a_{tj} \cdot v_j), \quad (2)$$

where $f_o : \mathbb{R}^d \to \mathbb{R}^d$ is a linear transformation, $c_{ti}, a_{ti} \in \mathbb{R}$ and $o_t \in \mathbb{R}^d$. Note that this is a fully *content-based attention* because it does not explicitly use any position or distance-related information about the query or keys.

4.2. Relative Attention

As another baseline, we use the relative attention mechanism[4] as used in Dai et al. (2019). Effectively, a sinusoidal positional encoding (Vaswani et al., 2017; Dai et al., 2019) is first used to create embeddings for each relative distance. Let $pe_k \in \mathbb{R}^d$ represent the embedding for the distance $k \in \mathbb{Z}$. Then, the relative position attention creates a query-key score sequence based on the corresponding relative distances between the query and the keys:

$$r_{ti} = \frac{< (q_t + b_2), pe_{i-t} >}{\sqrt{d}} \quad (3)$$

where $b_2 \in \mathbb{R}^d$ is a bias for position attention and $r_{ti} \in \mathbb{R}$. This is integrated with content-based attention by modifying Eqn. 1 in §4.1 as:

$$c_{ti} = \frac{< (q_t + b_1), k_i >}{\sqrt{d}} + r_{ti} \quad (4)$$

$b_1 \in \mathbb{R}^d$ is a bias for content-based attention. Everything else is kept the same as was for content-based attention.

4.3. Location Attention

Location attention, as introduced in Dubois et al. (2020), is primarily a form of attention based only on the positions of the encodings $e_{1:s}$; however, it is more expressive than the relative positional attention. Here we discuss the details of location attention with some refinements. Dubois et al. (2020) introduced a method to determine the locational "center of focus" for attention which is made to resemble human attention in visual search in how even when it focuses on a specific part of the input, it also perceives neighboring parts due to the eccentricity effect (Carrasco et al., 1995). Let $\mu_t \in \mathbb{R}$ represent the center of focus such that positions close to μ_t get higher attention than those farther away. With such μ_t, an attention spread can be modeled by using μ_t as a mean in a Gaussian distribution:

$$\lambda_{ti} = \exp\left(-\frac{(i - \mu_t)^2}{2 \cdot \sigma_t^2}\right) \quad (5)$$

Here σ_t is the standard deviation, which determines the spread of the attention focus. However, using raw values of i and μ_t is not ideal because the range of values (especially

of i) can differ based on sequence length. This becomes more problematic for unseen length generalization. Thus, the formalism is modified as follows:

$$\lambda_{ti} = \exp\left(-\frac{(norm(i) - clamp(\mu_t))^2}{2 \cdot \sigma_t^2}\right) \quad (6)$$

where:

$$norm(i) = \frac{i - 1}{max(1, s - 1)} \quad (7)$$

$$clamp(\mu_t) = max(0 + m \cdot \mu_t, min(1 + m \cdot \mu_t, \mu_t)) \quad (8)$$

Note that the encoder position index ranges from 1 to s where s is the sequence length. The $norm()$ function squeezes any position index i into the range $[0, 1]$ no matter the sequence length. Further, the $clamp()$ function enforces μ_t to be approximately within $[0, 1]$ which is the possible range of positions that can be attended. Following Dubois et al. (2020), m in $clamp()$ acts as a negative slope ($m = 0.01$) to add some "leakiness" similar to LeakyReLU. Note that the result is a PDF over the whole real number set whereas only the discrete positions of the encoding matter. Thus, λ_{ti} can be further normalized to get a discretized probability measure over only the relevant positions:

$$\lambda'_{ti} = \frac{\lambda_{ti}}{\sum_{j=1}^{s} \lambda_{tj}} \quad (9)$$

This gives the location attention. Below we discuss how to obtain μ_t and σ_t. First, a transformation over the decoder hidden state h_t is created as $l_t = f_l(h_t)$ where $f_l : \mathbb{R}^d \to \mathbb{R}^d$ is a linear transformation.[5] Next, σ_t is computed as:

$$\sigma_t = \frac{ReLU(f_{\sigma_t}(l_t)) + min_{\sigma_t}}{s} \quad (10)$$

Here $f_{\sigma_t} : \mathbb{R}^d \to \mathbb{R}$ is a linear transform and min_{σ_t} is the minimum value for σ_t. Next, μ_t is computed by taking some steps (in either forward or backward direction) with respect to some reference position ref_t. Given that the $norm(.)$ function will squeeze any discrete position index into a continuous number in $[0, 1]$, ref_t can also be treated to be in $[0, 1]$. Formally, μ_t is computed as:

$$\mu_t = \text{ref}_t + \text{stepsize} \cdot \text{steps}_t \quad (11)$$

Here, stepsize is $\frac{1}{max(1, s-1)}$. For the reference point ref_t, different possible choices can be considered. One possible choice is the previous attended position pa_{t-1} which is computed as $pa_{t-1} = \sum_{i=1}^{s} \alpha_{t-1i} \cdot norm(i)$ where α_{t-1i} represents the interlayer attention at the previous timestep $(t - 1)$ to the encoding position i. Essentially, with this setup, the attention model can move left or right with respect to previously attended position. Another choice for the

---

[4]Initially the idea was introduced in Shaw et al. (2018).

[5]Dubois et al. (2020) used a GRU for f_l. However, in our experiments we removed it because we did not find it effective.

reference point is to simply make a neural network-based logistic prediction to choose any arbitrary position b_t in $[0, 1]$ as $b_t = sigmoid(f_b(l_t))$ where $f_b : \mathbb{R}^d \to \mathbb{R}$ is a linear transform. Here b_t can also help initialize ref_t to adaptively learn to start attending from the beginning of the encoding ($i = 1$) or the end of the encoding ($i = s$) (or even somewhere in-between if needed) based on the task. Ultimately, we can allow the model itself to learn to choose or combine both pa_{t-1} and b_t as needed:

$$\text{ref}_t = g_t \cdot pa_{t-1} + b_t \tag{12}$$

with g_t being a real scalar in $[0, 1]$ functioning as a gate that decides to keep or ignore pa_{t-1}. It is computed as $g_t = sigmoid(f_g(l_t))$ where $f_g : \mathbb{R}^d \to \mathbb{R}$ is a linear transform. Next the steps to take (i.e., steps$_t$) with respect to the reference point are determined as follows:

$$\text{steps}_t = \text{softstair}(f_{step}(l_t)) \tag{13}$$

where $f_{step} : \mathbb{R}^d \to \mathbb{R}$ is again a linear transform and softstair is an activation function that pushes the output towards an integer:

$$\text{softstair}(x) = \lfloor x \rfloor + sigmoid(\tau \cdot (x - \lfloor x \rfloor - 0.5)) \tag{14}$$

τ is a temperature hyperparameter which is set to 20 like in Dubois et al. (2020). Last, the attention is computed as a convex combination of content attention and location attention:

$$a_{ti} = mix_{ti} \cdot \left(\frac{\exp(c_{ti})}{\sum_{j=1}^{s} \exp(c_{tj})} \right) + (1 - mix_{ti}) \cdot \lambda'_{ti} \tag{15}$$

$$mix_{ti} = sigmoid(\beta f_{mix}(h_t)) \tag{16}$$

Here $f_{mix} : \mathbb{R}^d \to \mathbb{R}$ is a linear transform and c_{ti} corresponds to the content-based pre-normalized attention scores as computed in Eqn. 1. In some cases, we might want to ignore the content attention focusing purely on location attention. In such cases, we can set $a_{ti} = \lambda'_{ti}$.

5. Proposed Approaches to Cross-Attention

In this section, we first present the limitations of the prior approaches discussed above and then present (in a *bottom-up* manner) our proposed changes that address them.

5.1. Limitations of Prior Approaches

Limitation 1 (handling reverse tasks): As noted earlier (see ReCopy task description in §2), in some tasks like Copy or Lookup, the target cross-attention position is always at the same constant relative distance from the timestep. In such cases, the inductive bias from the relative attention (§4.2) can be especially fruitful. However, this setup is not

maintained by default (without reversing the encoding or the input in the model), in the reverse directions of the tasks (Reverse Copy or Reverse Lookup). Consider transforming "4 7 9 8" to "8 9 7 4". In this case to print 8 in timestep $t = 1$, the model needs to attend to encoding position $i = 4$. Thus, the relative distance will be $i - t = 3$. However, for printing 9 in timestep $t = 2$, the model needs to attend to the encoding position $i = 3$. Then the relative distance will be $i - t = 1$. Thus, the ideal relative distance can vary with timestep and also depends on the source sequence length. These facts make it a struggle for relative attention, by default, to work on reverse tasks. In theory, location attention is equipped to handle reverse tasks - it has to just initialize b_t as 1 and g_t as 0 when $t = 1$. This will set $ref_t = 1$, i.e., the reference position will be the end of the input sequence. From that point location attention can learn to take steps backward one by one using previous attention (pa_{t-1}) as the reference position if needed. However, in practice, location attention still tends to be empirically brittle and have been shown to fail the reverse lookup task (Dubois et al., 2020).

Limitation 2 (handling ReCopy and beyond): As discussed in §2 (see ReCopy description), tasks like ReCopy, Reverse ReCopy, or their inverted variants are specifically designed to serve as settings in which the ideal attention position can vary from timestep to timestep (no matter if the encoding positions are reversed or not). Thus, this setting becomes hard for relative attention. Location attention, again, can theoretically address these situations given its flexibility to keep track of past attended position and ability to take any arbitrary steps in reference to past attended position dependent on the decoder state. Nevertheless, as mentioned earlier, in practice location attention turns out to be relatively brittle. Moreover, its use of soft sigmoid-based gating for making decisions at different stages of the model can lead to higher error accumulation and lower robustness to increasing lengths of data.

5.2. Bidirectional Relative Attention

First, we propose a simple approach to extend relative attention in a manner that addresses limitation 1. We note that if the task is, e.g., reverse copy, we can simply reverse the encoded sequence representations after encoding. Once done so, from the perspective of the decoder, the task becomes equivalent to forward copy. Nevertheless, in practice, we will not know ahead of time whether we are facing a task where forward version of the encoding is more ideal or the reversed version of the encoding. Thus, we use a gating mechanism that interpolates (make a convex combination of) the two directions of encoding so that the model can adaptively decide whether to reverse the encodings or not:

$$e_{1:s}^{rev} = \text{reverse}(e_{1:s}), \ \alpha_{dir} = \text{sigmoid}(\beta \cdot f_{dir}(e_{cls})) \tag{17}$$

$$\forall i \in \{1, \dots, s\} \quad e_i^{dir} = \alpha_{dir} \cdot e_i + (1 - \alpha_{dir}) \cdot e_i^{rev} \tag{18}$$

β is a scalar (acting as a temperature), $f_{dir} : \mathbb{R}^d \to \mathbb{R}^1$ is a linear layer, and $e_{cls} \in \mathbb{R}^d$ is a vector representation of the whole sequence $e_{1:s}$ - it can be implemented in multiple ways (we explain our implementation in Appendix E). After this we use the same strategy as in §4.2 but using key and value transformations over e^{dir} instead of e. This trick can also be useful in more practical tasks like translation where the source and target language may have different reading orders. Note that e^{dir} is different from outputs from models like bidirectional RNNs. Unlike here, in a bidirectional RNN, the encoded tokens remain in the same positions; only the contextualizing information comes from different directions. Also, note that this strategy is as general purpose as introducing bidrectionality to RNNs. Moreover, we are allowing neural networks to dynamically predict the direction for a given input through the gating mechanism; thus, avoiding infusion of task-specific knowledge of ideal direction of attention.[6]

5.3. OneStep Attention

As discussed in §5.1, fixing limitation 1 by reversing the encodings (as in §5.2) still does not address limitation 2. Concerned with limitation 2, we move away from simple relative positional attention and instead seek to make adjustments over location attention to address its potential issues (see §5.1). As a result we propose a new attention model - OneStep attention. Below we enumerate the main adjustments over location attention (from §4.3):

1. OneStep attends to key-value transformations of $e^{dir}_{1:s}$ instead of $e_{1:s}$ similar to §5.2.

2. The computation of ref_t is simplified as: $\text{ref}_t = pa_{t-1}$

3. The activation function in Eqn. 13 to sigmoid from softstair: $\text{steps}_t = \text{sigmoid}(f_{step}(l_t))$

First Change: The first change follows from §5.2 and is motivated to address limitation 1.

Second Change: The second change is motivated by a number of factors. First, due to the incorporation of the first change, the role of b_t from Eqn. 12 is severely undermined. It is not anymore necessary for b_t to initialize the starting position of attention to handle reverse tasks. Besides that the usefulness of b_t can be limited.[7] It is motivated for percentile attention in Dubois et al. (2020) which may not

be as relevant or can be accomodated by content attention mixing (Eqn. 15). So we removed it. To reduce error-accumulation we also remove the gating g_t over pa_{t-1}; thus ultimately setting $\text{ref}_t = pa_{t-1}$. It removes the models capacity for attending to some specific absolute position from the beginning/end but this capacity is also lacking from relative attention and is not currently required by most of our tasks. We keep investigation to incorporating this capacity better in the future. Currently, absolute positional encoding in the encoder combined with content attention mixing can still accommodate for the lack to an extent.

Third Change: In the third change, we replace softstair with a sigmoid for the step computation. The sigmoid function enforces the model to softly choose between either taking a single step forward ($\text{steps}_t = 1$) or none ($\text{steps}_t = 0$). We added this change because giving unrestricted freedom in determining the steps can make it harder for the model to learn the right function. Particularly in most of our current diagnostic tasks, it is sufficient to learn to make bounded steps in $[0, 1]$ with respect to the past attended position. While this choice is perhaps not ultimately ideal, it helps us evaluate the breaking points of the Location Attention framework better. Regardless, even after this restriction, OneStep can be still powerful enough to simulate a windowed version of relative attention (if it takes a single step in every timestep) (Shaw et al., 2018). Moreover, a sufficiently powerful encoded representation can, in theory, always reorganize or permute the input information to accommodate for this restriction. Besides, content attention mixing (Eqn. 15) can break the monotonicity of OneStep[8] and make it more flexible.

5.4. Monotonic Attention

In some tasks, it can be easier to learn to take bigger steps at the level of interlayer attention instead of expecting the encoder to permute the source input appropriately. So, we create another attention function where we relax the constraints in OneStep by changing the steps computation as:

$$\text{steps}_t = g \cdot \text{sigmoid}(f_{step}(l_t)) + (1-g) \cdot ReLU(f_{step}(l_t)) \tag{19}$$

Here, $g = \text{sigmoid}(p)$ where $p \in \mathbb{R}$ is a model parameter.[9] As we can see, with this setup we can allow the model itself to learn to prefer either taking controlled steps with a sigmoid or possibly bigger steps with a ReLU. We still use ReLU activation to keep the attention monotonic (i.e., the attention mechanism can only make forward steps) similar to OneStep for reasons discussed in §§5.3 (in Third Change).

---

[6]While this strategy may appear obvious, it is still not explored so far to our knowledge. Moreover, theoretical motivation does not always translate well to empirical performance. For instance, Location Attention struggles in reverse tasks despite having the theoretical capacity for reverse attention as discussed in §5.1. So empirical benefit of this strategy is not a priori obvious and deserves the investigation that we do here.

[7]It can be still useful in special cases when the model has to attend some x position from the end in one timestep and some y position from the beginning in another.

[8]By itself, without content attention mixing, OneStep is monotonic because in it, the center of focus can only move forward with time.

[9]In future, it can be better to have g dependent on the input encoding such as e^{cls} in case we want a multi-tasking model.

Model	Copy			Reverse Copy			Lookup			Reverse Lookup		
(Length Splits)	15	30	100	15	30	100	7	9	11	7	9	11
Content	0	0	0	0	0	0	33.3	0	0	3.7	0	0
Relative	**100**	**100**	**100**	0	0	0	**100**	**100**	**100**	78.2	0.8	0.4
LocAttn	99.8	0	0	0.7	0	0	**100**	9.4	0	13.3	0	0
Ours												
Bi-Relative	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
OneStepAttn	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
MonoAttn	**100**	**100**	**100**	**100**	**100**	**100**	**100**	98	29.9	28.5	0	0
Model	ReCopy			Reverse ReCopy			Inv ReCopy			Inv Reverse ReCopy		
(Length Splits)	15	30	100	15	30	100	15	30	100	15	30	100
Content	19.1	0	0	25	0	0	0.05	0	0	0	0	0
Relative	43.1	0	0	0.1	0	0	75.9	0	0	0	0	0
LocAttn	79.6	0	0	19.7	0	0	99.4	58.8	0	97.9	0.3	0
Ours												
Bi-Relative	33.4	0	0	35.3	0	0	69.8	0	0	71.3	0	0
OneStepAttn	**100**	**100**	**100**	**100**	**100**	**100**	0.1	0	0	0	0	0
MonoAttn	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**98.8**	**100**	**99.9**	**98.3**

Table 2. Accuracy of the models on different length generalization splits in different algorithmic diagnostic / probing tasks. We present the median of five runs on different seeds. We bold the best results.

Model	SCAN (Len.)	CFQ (Len.)
Content	17.61 ± 4.07	$\mathbf{62, 14 \pm 0.88}$
Relative	19.21 ± 5.52	56.64 ± 1.84
Mix LocAttn	20.74 ± 5.69	44.83 ± 9.45
Ours		
Bi-Relative	8.41 ± 1.21	59.48 ± 1.54
Mix OneStepAttn	$\mathbf{29.51 \pm 9.46}$	60.65 ± 3.74
Mix MonoAttn	21.08 ± 7.17	60.32 ± 3.58

Table 3. Accuracy on SCAN length split and CFQ length split. We report the mean and standard deviation of 5 runs for SCAN and of 3 runs for CFQ. We bold the best results.

6. Experimental Setup

Similar to Dubois et al. (2020), we use a Bidirectional GRU (Chung et al., 2014) based seq2seq model as the base for all the attention mechanisms. We explain more architectural details and hyperparameters in the Appendix E.

Nomenclature: In Tables 2 and 3, we use the term *Content* to refer to content attention (§4.1), *Relative* to refer to relative attention (§4.2), and *Bi-Relative* for bi-directional relative attention (§5.2). We use the terms *LocAttn*, *OneStepAttn*, and *MonoAttn* for location attention (§4.3), OneStep Attention (§5.3), and monotonic attention (§5.4) respectively if they are used without mixing content attention (i.e., replacing Eqn. 15 with $a_{ti} = \lambda'_{ti}$). Otherwise, we use the terms *Mix LocAttn*, *Mix OneStepAttn*, and *Mix MonoAttn* when mixing with content attention is done (i.e., Eqn. 15 is kept as described). We generally use the unmixed variants on the simpler diagnostic tasks (Lookup, Copy, or ReCopy-based tasks) because position-based attention is what is mainly relevant for the tasks.

Evaluation: We calculate the sequence-level accuracy of our models. Any generated output gets a score of 1 if and only if it matches exactly with the given target output.

On the EOS problem: The EOS token is a special marker that a model needs to generate to signify the end of sequence. In similar contexts, some prior works have tried to make the evaluation less stringent (Dubois et al., 2020; Newman et al., 2020) by terminating the model generation based on the oracle EOS position or by truncating oracle sequence based on predicted EOS position. We do not modify the evaluation in any such non-standard manner. Generally, we do not find EOS prediction to be a problem. If the inductive bias is suitable for the task, our models learn to generalize near perfectly without us needing to incorporate any separate mechanism to predict EOS properly.

7. Experimental Results

In Table 2 we show the results of our different attention strategies on all our diagnostic tasks except SCAN and CFQ. The results are close to what we would expect a priori. Pure content attention (Content) without more explicit guidance from any positional information suffers in all the tasks. Relative attention (Relative) does well in the forward copy and lookup tasks but it fails in the reversed tasks for the reasons discussed in §5.2. It also fails in the ReCopy-based tasks. This is consistent with our discussed limitations of prior works in §5.1. Also, consistent with this discussion, we find our implementation of location attention (LocAttn) to struggle in all the tasks.

Bidirectional relative attention (Bi-Relative) succeeds on

Model	Copy			Reverse Copy			Lookup			Reverse Lookup		
(Length Splits)	15	30	100	15	30	100	7	9	11	7	9	11
OneStepAttn	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
−Step 2	**100**	1.4	0	**100**	98.6	0	99.2	2.34	0	99.8	0	0
−Step 3	6.9	0	0	0	0	0	41.9	0	0	22.9	0	0
−Sigmoid	**100**	**100**	99.8	**100**	**100**	**100**	**100**	74.3	0.3	19.1	0	0

Model	ReCopy			Reverse ReCopy			Inv ReCopy			Inv Reverse ReCopy		
(Length Splits)	15	30	100	15	30	100	15	30	100	15	30	100
OneStepAttn	**100**	**100**	**100**	**100**	**100**	**100**	0.1	0	0	0	0	0
−Step 2	15.9	0	0	16.9	0	0	**95.5**	0	0	**96.2**	0	0
−Step 3	**100**	**100**	**100**	**100**	**100**	99.9	40.3	0	0	45	0	0
−Sigmoid	**100**	**100**	**100**	**100**	**100**	**100**	0	0	0	22	0	0

Table 4. Accuracy of ablations over OneStepAttn in different length generalization splits in different algorithmic diagnostic/probing tasks. We present the median of five runs on different seeds. We bold the best results.

both forward and reverse directions of copy and lookup tasks. This is aligned with our motivation for designing it (§5.2). However, Bidirectional relative attention still does not alleviate the second limitation (§5.1) and thus, fail in the ReCopy-based tasks.

OneStep attention (OneStepAttn) succeeds nearly on all tasks except the inverted variations of the ReCopy tasks. The Copy tasks and Lookup tasks are easy to learn for OneStep attention because in either tasks it has to simply learn to take one step forward relative to the past attended position in every timestep. The ReCopy and Reverse ReCopy is slightly more complicated but still not too hard to learn. In these cases, the model has to learn to wait (predict $steps_t = 0$) while the decoder is repeating previous generations. The attention model has to then predict $steps_t = 1$ to move one step forward in the encoding positions after the repetition of the content from the past attended position is complete. Thus, the OneStep strategy is suitable for the ReCopy and Reverse ReCopy tasks as well.

However, the OneStep strategy faces an issue for the inverted versions of the tasks. Consider an Inv ReCopy sample where the input is "4 4 4 7 7 7 7 7 9 9 9 9 9 8 8 8 8 8" and the output is "4 7 9 8". In this case, one way to solve this would be for the encoder to radically re-organize the positions of the input information. But if the encoder fails to do that and keeps the encoded information close to its original position, OneStep attention, by itself, is ill-equipped for the task. In the given example, after printing 4 from encoding position 1, in the next timestep it has to take not just one but three steps forward. OneStep attention cannot do that because its number of steps is constrained by a sigmoid.

In contrast to OneStep attention, monotonic attention (MonoAttn) is more flexible allowing bigger steps when needed. As such, monotonic attention is able to solve Inv ReCopy tasks that OneStep could not. It also performs perfectly on copy tasks and ReCopy tasks in both direc-

tions. However, it fails on the lookup tasks. It seems that its increased flexibility (loosened inductive bias) and its possibility to make more uncontrolled steps (which are unnecessary for the lookup tasks) also at the same time make it more confused when trying to learn the lookup tasks in a length-generalizing manner.

Ultimately, both OneStep attention and monotonic attention perform better than any of the other attention models. Both solves 6 out of the 8 tasks in Table 2 with 100% accuracy. However, we also discover a trade-off - the restricted steps of OneStep attention preclude it from solving the inverted versions of ReCopy tasks whereas the more unconstrained steps of monotonic attention manages the inverted ReCopy tasks but at the cost of the lookup tasks.

In Table 3, we present the results on SCAN. We find location attention and our extensions of it (OneStep attention or monotonic attention) to generally also perform better on the task of translating simple commands into sequences of actions than other forms of interlayer attention even though they are not designed explicitly keeping the structure of SCAN task in mind. OneStep attention (Mix OneStepAttn) does particularly better than the others in SCAN. In the same table, we also present the results on CFQ. Interestingly, the basic position-encoding-less version of inter-layer attention does the best here. However, both OneStep and monotonic attention keep up with it better than others - like location attention or unidirectional relative attention.

7.1. Additional Analyses

Ablations: In Table 4, we show some of the main ablations of OneStep Attention. −Step 2 represents using the more sophisticated location attention variant of ref_t computation (Eqn. 12) instead of the proposed $ref_t = pa_{t-1}$ change in step 2 in §5.3. −Step 3 represents using softstair activation for step computation (Eqn. 13) from location attention instead of the proposed sigmoid activation in step 3 change of

OneStep (§5.3). −Sigmoid represents removing the activation function from Eqn. 13 altogether. As the ablation shows both of our proposed changes are important to succeed in most of the tasks. Interestingly, we find here that having no activation at all in Eqn. 13 generally serves us better than having softstair activation. Besides that, the ablation results support our original motivation for proposing Step 2 and Step 3 in OneStep attention. We show several more ablation tests in Appendix B.

Additional Tasks: In Appendix A, we introduce and explore two additional tasks - **DeDupe** and **PosRetrieve**.

Alternate Evaluation: In Appendix C we evaluate the models on edit distance instead of exact match accuracy. Edit distance serves as a more fine-grained evaluation.

Examples: In Appendix D we present some example failure cases of OneStep attention and monotonic attention.

8. Limitations

First, although OneStepAttn and MonoAttn perform better than LocAttn in general, they are also more restricted. Nevertheless, OneStepAttn and MonoAttn show the potential of the LocAttn-like framework with restrained room for error accumulation and slightly stronger inductive biases. Ideally, we want to improve upon them in the future to get both higher flexibility and good performance. Moreover, when building specific modelling capacities (say attending to absolute positions), we should also consider building appropriate synthetic tasks for sanity checking in a similar spirit as done in this paper. In Appendix A, we propose PosRetrieve which can be a sanity check for absolute position attention capability for future developments.

Second, our experiments are limited to mainly synthetic tasks most of which require purely location-based attention[10] but no complex synergy between content-based attention and position-based attention. More synthetic tasks for sanity checking such capacities can be built.

Third, our exploration is currently limited to RNN-based seq2seq models. One reason for focusing on RNNs is because vanilla non-pretrained Transformers encoders can struggle to solve tasks like lookup table for decoder to do its job without specific changes (Csordás et al., 2022). Moreover, integration of location attention into Transformers is complicated by the fact that they use multiple layers of cross-attention in each timestep introducing additional variables to consider (the problem is not that our methods cannot be integrated with Transformers but that there are many ways to do so). Given these added variables, we leave investigations

---

[10]Although, we should note that despite their simplicity, the tasks still have been difficult to solve perfectly (Dubois et al., 2020; Dehghani et al., 2019; Liang et al., 2021)

with Transformers for future work.

9. Conclusion

We introduce several new probing tasks - ReCopy and its variants (some others in Appendix A) to enable additional diagnoses of length generalization performance of neural models. Although our proposed tasks are simple, this very simplicity can allow better isolation of failure cases and provide sanity checks for locational reasoning skills. Moreover, the new tasks are still challenging enough that none of the models explored here succeed in all of them.

We propose a way to softly switch between the forward encodings and its reversed version to get near perfect performance in reverse variants of copy and lookup tasks that have been previously challenging to solve. We illuminate the limits of location attention and show how certain modifications in the form of OneStep attention and monotonic attention can bring massive improvement. Although, the modifications bring stronger inductive biases than location attention, they can still simulate windowed relative attention and empirically demonstrate more stable performance across datasets including more realistic ones like CFQ.

Monotonic attention or OneStep attention can also be more broadly applicable in any context requiring *list traversal* i.e. monotonic traversal through a list of items in a backpropagation-friendly manner — for example, one application can be skill selection with a dynamic time horizon instead of a fixed one (Garg et al., 2022). OneStep attention is suitable if the only relevant choice during the traversal is to either stay at a position or move to the next position by a single step. Monotonic attention is suitable if we also want to allow the model to skip positions during traversal.

References

Anil, C., Wu, Y., Andreassen, A. J., Lewkowycz, A., Misra, V., Ramasesh, V. V., Slone, A., Gur-Ari, G., Dyer, E., and Neyshabur, B. Exploring length generalization in large language models. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=zSkYVeX7bC4.

Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. In Bengio, Y. and LeCun, Y. (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1409.0473.

Bai, S., Kolter, J. Z., and Koltun, V. Deep equilibrium models. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/01386bd6d8e091c2ab4c7c7de644d37b-Paper.pdf.

Banino, A., Balaguer, J., and Blundell, C. Pondernet: Learning to ponder. *ICML Workshop*, abs/2107.05407, 2021. URL https://arxiv.org/abs/2107.05407.

Bird, S., Klein, E., and Loper, E. *Natural Language Processing with Python*. O'Reilly Media, Inc., 1st edition, 2009. ISBN 0596516495.

Carrasco, M., Evert, D. L., Chang, I., and Katz, S. M. The eccentricity effect: Target eccentricity affects performance on conjunction searches. *Perception & Psychophysics*, 57(8):1241–1261, November 1995. doi: 10.3758/bf03208380. URL https://doi.org/10.3758/bf03208380.

Chang, T., Xu, Y., Xu, W., and Tu, Z. Convolutions and self-attention: Re-interpreting relative positions in pre-trained language models. 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. 4322–4333, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.333. URL https://aclanthology.org/2021.acl-long.333.

Chung, J., Gülçehre, Ç., Cho, K., and Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *NeurIPS Workshop*, abs/1412.3555, 2014. URL http://arxiv.org/abs/1412.3555.

Csordás, R., Irie, K., and Schmidhuber, J. The neural data router: Adaptive control flow in transformers improves systematic generalization. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=KBQP4A_J1K.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q., and Salakhutdinov, R. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2978–2988, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1285. URL https://aclanthology.org/P19-1285.

Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J., and Kaiser, L. Universal transformers. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HyzdRiR9Y7.

Dubois, Y., Dagan, G., Hupkes, D., and Bruni, E. Location Attention for Extrapolation to Longer Sequences. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 403–413, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.39. URL https://aclanthology.org/2020.acl-main.39.

Dufter, P., Schmitt, M., and Schütze, H. Position Information in Transformers: An Overview. *Computational Linguistics*, 48(3):733–763, 09 2022. ISSN 0891-2017. doi: 10.1162/coli_a_00445. URL https://doi.org/10.1162/coli_a_00445.

Garg, D., Vaidyanath, S., Kim, K., Song, J., and Ermon, S. LISA: Learning interpretable skill abstractions from language. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=XZhipvOUBB.

Graves, A. Adaptive computation time for recurrent neural networks. *ArXiv*, abs/1603.08983, 2016. URL http://arxiv.org/abs/1603.08983.

Graves, A., Wayne, G., and Danihelka, I. Neural turing machines. *ArXiv*, abs/1410.5401, 2014. URL http://arxiv.org/abs/1410.5401.

Kaiser, L. and Sutskever, I. Neural gpus learn algorithms. In *ICLR (Poster)*, 2016. URL http://arxiv.org/abs/1511.08228.

Ke, G., He, D., and Liu, T.-Y. Rethinking positional encoding in language pre-training. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=09-528y2Fgf.

Keysers, D., Schärli, N., Scales, N., Buisman, H., Furrer, D., Kashubin, S., Momchev, N., Sinopalnikov, D., Stafiniak, L., Tihon, T., Tsarkov, D., Wang, X., van Zee, M., and Bousquet, O. Measuring compositional generalization: A comprehensive method on realistic data. In *International Conference on Learning Representations*,

2020. URL https://openreview.net/forum?id=SygcCnNKwr.

Kim, N., Linzen, T., and Smolensky, P. Uncontrolled lexical exposure leads to overestimation of compositional generalization in pretrained models, 2022. URL https://arxiv.org/abs/2212.10769.

Lake, B. and Baroni, M. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2873–2882. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/lake18a.html.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7871–7880, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.703. URL https://aclanthology.org/2020.acl-main.703.

Liang, K.-P., Anil, C., Wu, Y., and Grosse, R. B. Out-of-distribution generalization with deep equilibrium models. In *ICML Workshop*, 2021.

Likhomanenko, T., Xu, Q., Synnaeve, G., Collobert, R., and Rogozhnikov, A. Cape: Encoding relative positions with continuous augmented positional embeddings. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 16079–16092. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/865bf46435bd84fa5d89f64cf3ba7347-Paper.pdf.

Liska, A., Kruszewski, G., and Baroni, M. Memorize or generalize? searching for a compositional RNN in a haystack. *ICML Workshop*, abs/1802.06467, 2018. URL http://arxiv.org/abs/1802.06467.

Luo, S., Li, S., Cai, T., He, D., Peng, D., Zheng, S., Ke, G., Wang, L., and Liu, T.-Y. Stable, fast and accurate: Kernelized attention with relative positional encoding. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 22795–22807. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/c0f168ce8900fa56e57789e2a2f2c9d0-Paper.pdf.

Luo, S., Li, S., Zheng, S., Liu, T.-Y., Wang, L., and He, D. Your transformer may not be as powerful as you expect. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=NQFFNdsOGD.

Luong, T., Pham, H., and Manning, C. D. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1412–1421, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1166. URL https://aclanthology.org/D15-1166.

Newman, B., Hewitt, J., Liang, P., and Manning, C. D. The eos decision and length extrapolation. In *BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, 2020.

Nye, M. I., Andreassen, A. J., Gur-Ari, G., Michalewski, H., Austin, J., Bieber, D., Dohan, D., Lewkowycz, A., Bosma, M., Luan, D., Sutton, C., and Odena, A. Show your work: Scratchpads for intermediate computation with language models. *ArXiv*, abs/2112.00114, 2021. URL https://arxiv.org/abs/2112.00114.

Press, O., Smith, N., and Lewis, M. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=R8sQPpGCv0.

Qu, A., Niu, J., and Mo, S. Explore better relative position embeddings from encoding perspective for transformer models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 2989–2997, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.237. URL https://aclanthology.org/2021.emnlp-main.237.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

Schmidhuber, J. Self-delimiting neural networks. *arXiv*, abs/1210.0118, 2012. URL http://arxiv.org/abs/1210.0118.

Shaw, P., Uszkoreit, J., and Vaswani, A. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pp. 464–468, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2074. URL https://aclanthology.org/N18-2074.

Su, J., Lu, Y., Pan, S., Wen, B., and Liu, Y. Roformer: Enhanced transformer with rotary position embedding. *ArXiv*, abs/2104.09864, 2021. URL https://arxiv.org/abs/2104.09864.

Sun, Y., Dong, L., Patra, B., Ma, S., Huang, S., Benhaim, A., Chaudhary, V., Song, X., and Wei, F. A length-extrapolatable transformer. *arXiv preprint arXiv:2212.10554*, 2022.

Sutskever, I., Vinyals, O., and Le, Q. V. Sequence to sequence learning with neural networks. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K. (eds.), *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper/2014/file/a14ac55a4f27472c5d894ec1c3c743d2-Paper.pdf.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Wang, B., Zhao, D., Lioma, C., Li, Q., Zhang, P., and Simonsen, J. G. Encoding word order in complex embeddings. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=Hke-WTVtwr.

Wennberg, U. and Henter, G. E. The case for translation-invariant self-attention in transformer-based language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pp. 130–140, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-short.18. URL https://aclanthology.org/2021.acl-short.18.

Wu, C., Wu, F., and Huang, Y. DA-transformer: Distance-aware transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2059–2068, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.166. URL https://aclanthology.org/2021.naacl-main.166.

Wu, F., Fan, A., Baevski, A., Dauphin, Y., and Auli, M. Pay less attention with lightweight and dynamic convolutions. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SkVhlh09tX.

Yang, B., Tu, Z., Wong, D. F., Meng, F., Chao, L. S., and Zhang, T. Modeling localness for self-attention networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 4449–4458, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1475. URL https://aclanthology.org/D18-1475.

A. Additional Tasks

We introduce and analyze two additional tasks here. We present examples for each in Table 5.

DeDupe: As the name suggests, DeDupe is essentially a de-duplication task of removing contiguous repetitions[11] as shown in Table 5. The task is very similar to Inv ReCopy but with a few differences. In Inv ReCopy, repetition occurs based on a fixed rule, each number will be repeated some fixed number of times dependent on that specific number. The task in Inv ReCopy is to remove the fixed number of number-specific repetitions rather than all contiguous repetitions. For instance, if the input is "4 4 4 4 4 4" the output for Inv ReCopy will be "4 4" (because "4" is bound to repeat three times according to the rules of ReCopy) but for the same input, the output for DeDupe will be just "4". Unlike Inv ReCopy, the DeDupe function is not invertible. Thus the task cannot be inverted (whereas ReCopy is the inversion of Inv ReCopy). We generate the splits for this task (DeDupe) in the same way as the Copy task.

PosRetrieve: The task of PosRetrieve is to treat the input values as position indices to retrieve from. Given an input x in a list format, $x = [i, j]$, the output y will be $y = i$: $f(i, x); j : f(j, x);$. Here, $f(i, x) = x[i]$ if $len(x) > i$ else $f(i, x) = n/a$. We generate the splits for this task in the same way as the Copy task. This task can more explicitly check for a models ability to choose some p^{th} position item from the beginning. This was one ability for which location attention was motivated (Dubois et al., 2020), but there was no benchmark to explicitly check for this.

We mainly focus on the forward variations of the task here but reverse variants can also be created.

Results: In Table 6, we show the results of the main models on DeDupe and PosRetrieve. Only MonoAttn performs decently in DeDupe which makes sense given that it is the only approach that does well in Inv ReCopy which is similar to DeDupe. DeDupe is, however, a bit harder than Inv ReCopy for MonoAttn because the encoder needs to encode information about total contiguous repetitions from the context so that MonoAttn can predict the right amount of steps without looking ahead (which it cannot without some kind of multi-step attention). In Inv ReCopy, the encoder does not have to encode any contextual information, since repetition happens according to a fixed context-independent rule. Thus, we see reduced performance in DeDupe compared to that in Inv ReCopy. None of the models is currently able to do well at PosRetrieve. This is, perhaps, expected for most models since they are currently lacking any explicit capability for modelling absolute positions but Location Attention still struggles with it despite theoretically having the capacity. We leave this task as an open challenge.

---

[11]We thank one of our reviewers for this idea.

B. Ablations

B.1. Ablation Models and Other Alternatives

We also show experimental results of different potential alternatives in the vicinity of location attentions and ablations of monotonic attention. We discuss the different models below.

Bi-ROPE: Here, we use the same strategy as in §5.2 to create the encoder representations but then we apply a different positional encoding for modeling relative distances - rotary positional encodings (ROPE) (Su et al., 2021). ROPE rotates the query and key vectors in space based on their sinusoidally encoded positions before using the query and key in a content-based attention as in §4.1.

LocAttn S: This is a simplified (S) version of location attention (without content attention mixing) where we set $\text{ref}_t = b_t$ for the first timestep to initialize the reference position using b_t and then use $\text{ref}_t = pa_{t-1}$ like OneStep Attention/monotonic attention. This approach can be thought to be "in between" the original location attention and monotonic attention.

Mix LocAttn S: This is same as LocAttn S but with content attention mixing (Eqn. 15).

Mix LocAttn S PR: When mixing with content attention (Eqn. 15), there is an option to set pa_{t-1} to track only the location-based attended position to keep as a reference point by setting $pa_{t-1} = \sum_{i=1}^{s} \lambda'_{t-1i} \cdot norm(i)$ where λ'_{t-1i} is the location-only attention from the past timestep. We use the modifier PR (**P**osition Attention based **R**eference) to denote this way of setting pa_{t-1}. Mix LocAttn S PR extends Mix LocAttn S with PR.

Mix OneStepAttn PR: This extends Mix OneStepAttn with PR.

Mix MonoAttn PR: This extends Mix MonoAttn with PR.

RMonoAttn: In RMonoAttn (Relaxed Monotonic Attention) we remove the sigmoid completely and overall simplify the step computation to: $\text{steps}_t = ReLU(f_{step}(l_t))$.

Mix RMonoAttn: This is RMonoAttn with content attention mixing (Eqn. 15).

Mix MonoAttn PR: This extends Mix RMonoAttn with PR.

B.2. Ablation Results

In Table 7, we show the results of the above models in all the main paper tasks but SCAN and CFQ. Bi-ROPE performs similarly to Bi-Relative as we would expect. LocAttn S with its simplification performs better than LocAttn (Table 2) but still falls behind OneStep/monotonic attention. The Mix variant models tend to perform worse than the unmixed

Task	Input	Output
DeDupe	4 4 4 7 7 7 7 9 9 9 9 9 8 8 8 8 8	4 7 9 8
PosRetrieve	5 4 2 7 9 6 9 5 7 3	5:6; 4:9; 2:2; 7:5; 9:3; 6:9; 9:3; 5:6; 7:5; 3:7;

Table 5. Input-output examples for DeDupe and PosRetrieve

Model	DeDupe				PosRetrieve			
(Length Splits)	IID	15	30	100	IID	15	30	100
Content	98.7	37.1	0	0	**100**	0	0	0
Bi-Relative	**99.9**	68	0	0	**100**	0	0	0
LocAttn	99.6	96.9	71.4	0	96.3	0	0	0
OneStepAttn	94.1	51.	0	0	70.6	0	0	0
MonoAttn	99.7	**98.3**	**94.3**	**75**	79	0	0	0

Table 6. Accuracy of the models on different length generalization splits in DeDupe and PosRetrieve. We present the median of five runs on different seeds. We bold the best results.

.

ones - this is because these tasks can be done purely based on positional reasoning and the content attention is more likely to confuse the models than help in these specific tasks. PR can help better track past locationally attended positions and thus improve the performance of the Mix models (compared to when they are used without PR). RMonoAttn tends to struggle more compared to MonoAttn demonstrating the value of gating with sigmoid-activated step prediction (eqn. 19).

In Table 8, we show the results of the above models for SCAN. In SCAN, the trend reverses a bit - mix models tend to be here better than unmixed ones. We suspect that mixing with content attention is more beneficial in more sophisticated tasks (SCAN is at least relatively more sophisticated than others besides CFQ) because it adds more flexibility. PR can sometimes further help in SCAN too in some models.

C. Alternate Evaluations

In Table 9, we show the results of the main models on the probing tasks with mean edit distance[12] as the evaluation metric. This paints a more fine-grained picture of the differences between model performances.

D. Examples

In Table 10, we show some failure case examples from OneStep attention and monotonic attention. Generally we find that they can generate the sequence correctly from the beginning up to a point (the correct generated part is highlighted in cyan) after which things go awry.

---

[12]Computed using NLTK (Bird et al., 2009).

E. Experimental Details

E.1. Architecture

For the encoder, a stack of bi-directional GRUs is used. e_{cls} is the concatenation of the final hidden state of the forward encoder GRU and the first hidden state of the backward encoder GRU. In every timestep t, the decoder state h_{t-1} (h_0 initialized with e_{cls}) is used as a query and the encoded representations ($e_{1:s}$ or $e_{1:s}^{dir}$ depending on need) are used as keys and values. The values are passed through an extra non-linear transformation with LeakyRELU following the code from (Dubois et al., 2020) before the standard linear transformation. For the non-linear transformation we use the same d neurons per layer. The output of the attention is concatenated with the embedding of the last generated token (initially some special token "go" indicating start of sequence). The concatenation is used as input to a decoder stack of GRUs with h_{t-1} as the hidden memory. The output of decoder GRU is h_t. A linear transformation over h_t is used to change the size of the vector from d (size of h_t) to d_e (where d_e is the embedding size). We then use the transpose of the embedding matrix to create the distribution over vocabulary. We select the maximum scoring token as the generated token for the timestep t.

E.2. Hyperparameters

We attempted to keep the hyperparameters similar to (Dubois et al., 2020). We use 64 as the embedding size (i.e $d_e = 64$) and single-layered GRUs for the encoder/decoder. The total hidden size for the encoder/decoder GRU is 128 (therefore $d = 128$). We only use one head for the attention mechanism. We use a dropout of 50% on the encodings similar to Dubois et al. (2020). We set $\beta = 5$. Following tradition we keep attention head dimension as d/heads which in our case is 128 since heads $= 1$. For CFQ, we use two layered GRUs for the encoder/decoder and twice

Model	Copy			Reverse Copy			Lookup			Reverse Lookup		
(Length Splits)	15	30	100	15	30	100	7	9	11	7	9	11
Bi-ROPE	**100**	**100**	0	**100**	**100**	0	**100**	**100**	**100**	89.5	1.6	0.2
Mix LocAttn	99.8	0	0	1.5	0	0	8.6	0	0	7.5	0	0
LocAttn S	**100**	26.6	0	**100**	74.1	0	**100**	46.3	0	99.6	0.1	0
Mix LocAttn S	99.7	0	0	85.9	0	0	33.5	0	0	3.5	0	0
Mix LocAttn S PR	83.6	0	0	99.7	10.5	0	32.9	0	0	1.1	0	0
Mix OneStepAttn	2.6	0	0	0.4	0	0	**100**	71.7	1.6	**100**	**79.1**	**8.1**
Mix OneStepAttn PR	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**47.8**	57.5	0	0
Mix MonoAttn	99.95	5	0	**100**	50.8	0	94.5	0	0	53.5	0	0
Mix MonoAttn PR	**100**	99.2	86.7	**100**	**100**	99.7	15.3	0	0	0	0	0
RMonoAttn	**100**	**100**	**100**	**100**	**100**	99.9	64.3	0	0	72.8	0	0
Mix RMonoAttn	31.1	0	0	**100**	66.3	0	99.8	0	0	41.4	0	0
Mix RMonoAttn PR	0	0	0	**100**	**100**	**100**	39.2	0	0	4.3	0	0
Model	ReCopy			Reverse ReCopy			Inv ReCopy			Inv Reverse ReCopy		
(Length Splits)	15	30	100	15	30	100	15	30	100	15	30	100
Bi-ROPE	43.4	0	0	39	0	0	56.0	0	0	53.1	0	0
Mix LocAttn	36.1	0	0	30.4	0	0	99.6	65.1	0	98.6	24.1	0
LocAttn S	98.7	5.2	0	99.9	1.4	0	100	99.3	91.1	99.9	99.3	91.8
Mix Location S	99.7	0	0	99.6	0	0	98.9	66.6	0	98.8	57.7	0
Mix Location S PR	99.6	0.4	0	**100**	61.4	0	99.8	98.6	88	99.6	98.1	84.8
Mix OneStepAttn	43.5	0	0	87.1	0	0	10.4	0	0	5.45	0	0
Mix OneStepAttn PR	99.9	88.4	30.2	**100**	**100**	**100**	0	0	0	0	0	0
Mix MonoAttn	7.09	0	0	99.85	0	0	99.8	82.4	0	**100**	82.2	0
Mix MonoAttn PR	**100**	**100**	**100**	**100**	53	0.5	0	0	0	89.6	77.6	42.1
RMonoAttn	**100**	**100**	99.9	0	0	0	100	100	99.1	100	100	99
Mix RMonoAttn	98.1	0	0	23.4	0	0	99.85	70.5	0	100	83.8	0
Mix RMonoAttn PR	**100**	90.8	51.2	**100**	**100**	**100**	0	0	0	40	0	0

Table 7. Accuracy of the models on different length generalization splits in different algorithmic diagnostic/probing tasks. We present the median of five runs on different seeds. We bold the best results.

Model	SCAN (Length Split)
Bi-ROPE	10.46 ± 3.78
LocAttn	24.56 ± 17.51
LocAttn S	25.12 ± 6.30
Mix LocAttn S	27.55 ± 10.10
Mix LocAttn S PR	19.8 ± 3.26
OneStepAttn	15.38 ± 0.42
Mix OneStepAttn PR	17.67 ± 3.54
MonoAttn	14.98 ± 0.15
Mix MonoAttn PR	26.68 ± 10.27
RMonoAttn	15.28 ± 1.47
Mix RMonoAttn	22.92 ± 6.39
Mix RMonoAttn PR	$\mathbf{27.99 \pm 11.26}$

Table 8. Accuracy on SCAN length split. We report the mean and std of 5 runs. We bold the best results.

the hidden size/embedding size than above. Generally, we use a batch size 32, a learning rate of $1e-3$ with Adam (default parameters) and no weight decay. We halve the learning rate if the accuracy plateaus for four contiguous epochs. We run the models for a maximum of 100 epochs with 50 patience for early stopping. More hyperparameter

details can be found in the codebase.

F. Connections to Other Models

Neural Turing Machines (Graves et al., 2014) include one of the earliest incorporation and motivation of a location-based attention (distinguished from content based attention) within a modern neural network-based paradigm. Luong et al. (2015) proposed a Gaussian distribution-based attention for localized focus which is similar to how eccentricity effect is modeled here. Location attention from Dubois et al. (2020) expands upon many of these prior ideas.

In the context of Transformers, countless works proposed ways to include some form of position-based attention bias (Shaw et al., 2018; Yang et al., 2018; Dai et al., 2019; Wang et al., 2020; Ke et al., 2021; Su et al., 2021; Luo et al., 2021; Qu et al., 2021; Chang et al., 2021; Wu et al., 2021; Wennberg & Henter, 2021; Likhomanenko et al., 2021; Dufter et al., 2022; Luo et al., 2022; Sun et al., 2022) (interalia). Dynamic convolution (Wu et al., 2019) and other similar models can also be treated as forms of location attention (query-to-distance-based attention within a local

Model	Copy			Reverse Copy			Lookup			Reverse Lookup		
(Length Splits)	15	30	100	15	30	100	7	9	11	7	9	11
Content	6.1	21	90.7	6.8	21.9	91.7	1.4	4.6	6.5	1	3.8	5.4
Relative	**0**	**0**	**0**	4.4	20.7	90.2	**0**	**0**	**0**	0.3	4.9	6.3
LocAttn	**0**	14.2	82	1.7	16.7	84.9	1.2	3	5.7	1.3	3.9	6.2
Ours												
Bi-Relative	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
OneStepAttn	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
MonoAttn	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	0.9	0.7	3	4.9
Model	ReCopy			Reverse ReCopy			Inv ReCopy			Inv Reverse ReCopy		
(Length Splits)	15	30	100	15	30	100	15	30	100	15	30	100
Content	8	53.4	249.2	8.1	53.7	249.3	4	19.7	90.1	4.3	20.1	90.4
Relative	3.7	31.5	160	17.9	60.8	250.7	0.5	13.7	73.8	3.2	18.4	88.3
LocAttn	0.8	56.5	233	2.7	39.1	226.1	**0**	0.5	45.8	**0**	7.8	72.5
Ours												
Bi-Relative	4.5	32.4	159.6	4.2	31.9	159.1	0.7	14.4	74	0.6	13.9	74
OneStepAttn	**0**	**0**	**0**	**0**	**0**	**0**	3	16.6	86	3.1	17.2	85.7
MonoAttn	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Table 9. Average edit distance (lower the better) of the models on different length generalization splits in different algorithmic diagnostic/probing tasks. We present the median of five runs on different seeds. We bold the best results.

Model	Task	Examples
OneStepAttn	Inv ReCopy	**Input:** 1 2 5 5 5 5 5 5 7 7 7 7 7 2 9 9 9 9 9 9 9 9 9 3 7 7 7 7 7 9 9 9 9 9 4 4 4 5 5 5 3 3 4 4 4 0 1 1 4 4 4 **Oracle:** 1 2 5 5 7 2 9 9 3 7 9 4 5 3 3 4 0 1 1 4 **Prediction:** 1 2 5 5 7 2 9 9 3 9 7 3 4 3
OneStepAttn	Inv ReCopy	**Input:** 9 9 9 9 9 2 1 1 6 6 6 6 6 6 3 7 7 7 7 7 6 6 6 8 8 8 8 8 1 5 5 5 4 4 4 3 7 7 7 7 7 4 4 4 5 5 5 7 7 7 7 7 3 7 7 7 7 7 **Oracle:** 9 2 1 1 6 6 3 7 6 8 1 5 4 3 7 4 5 7 3 7 **Prediction:** 9 2 1 1 6 6 3 7 6 8 1 4 5
MonoAttn	Reverse Lookup	**Input:** t4 t4 t1 t6 t5 t2 t6 t3 t4 100 . **Oracle:** 100 001 100 010 011 010 111 111 101 110 **Prediction:** 100 001 100 010 110 100
MonoAttn	Reverse Lookup	**Input:** t4 t4 t2 t6 t5 t4 t2 t2 t3 110 . **Oracle:** 110 010 011 101 110 011 001 100 001 100 **Prediction:** 110 010 011 101 110

Table 10. Failure case examples of OneStepAttn and MonoAttn. We highlight the matching subsequence in cyan.

window). Most of the approaches mentioned in this paragraph, however, involve emphasis on relative distances. As we find in our investigations based on two popular representatives of such forms of attention — Relative attention (Dai et al., 2019) and ROPE (Su et al., 2021) — they tend to struggle on tasks like ReCopy where the ideal relative distance of attention position can be arbitrarily big and can vary with every timestep.

Press et al. (2022) introduced a new positional encoding technique for transformer-based decoders for better length generalization on language modeling. They add a bias towards locality for that purpose. However, the advantage of locality bias in the contexts of seq2seq tasks that we consider here are less clear given that the ideal position of attention can be arbitrarily distant from the current timestep

of decoding in our tasks. Transformers can iteratively transfer information depth-wise through local operations but that will be also limited by the maximum layer depth. However, allowing adaptive layers (Schmidhuber, 2012; Graves, 2016; Bai et al., 2019; Banino et al., 2021) or intermediate computation based on scratchpad (Nye et al., 2021) may mitigate these issues. Recently, Anil et al. (2022) showed that length generalization in large language models can be enhanced by a careful synergy of different techniques such as scratchpad and in-context examples.

Neural GPU (Kaiser & Sutskever, 2016) also achieves strong length generalization performance on several algorithmic tasks but with curriculum learning. Csordás et al. (2022) solve the lookup table tasks in both directions with gated Transformer-based models but it does so only at the

level of encoding (which can be already done with a Bidirectional RNN as shown in the same paper) where it only has to learn to compute and output the final function output. It does not tackle the challenge of doing the task at a seq2seq level (or in the style of a language model) which requires printing a sequence of intermediate function outputs in a rule-based manner in addition to the final output.