State Spaces Aren't Enough: Machine Translation Needs Attention

Ali Vardasbi[†*] **Telmo Pessoa Pires**[†] **Robin M. Schmidt** **Stephan Peitz**
University of Amsterdam Apple
a.vardasbi@uva.nl {telmo, robin_schmidt, speitz}@apple.com

Abstract

Structured State Spaces for Sequences (S4) is a recently proposed sequence model with successful applications in various tasks, e.g. vision, language modeling, and audio. Thanks to its mathematical formulation, it compresses its input to a single hidden state, and is able to capture long range dependencies while avoiding the need for an attention mechanism. In this work, we apply S4 to Machine Translation (MT), and evaluate several encoder-decoder variants on WMT'14 and WMT'16. In contrast with the success in language modeling, we find that S4 lags behind the Transformer by approximately 4 BLEU points, and that it counter-intuitively struggles with long sentences. Finally, we show that this gap is caused by S4's inability to summarize the full source sentence in a single hidden state, and show that we can close the gap by introducing an attention mechanism.

1 Introduction

The Transformer (Vaswani et al., 2017) is the most popular architecture for state-of-the-art Natural Language Processing (NLP) (Devlin et al., 2019; Brown et al., 2020; NLLB Team et al., 2022). However, the attention mechanism on which it is built is not well suited for capturing long-range dependencies due to its quadratic complexity (Ma et al., 2023). Recently, Structured State Spaces for Sequences (S4) was shown to be on par with the Transformer on various sequence modelling tasks, including time series forecasting, language modeling (Gu et al., 2022), and audio generation (Goel et al., 2022); and to surpass the Transformer on tasks requiring reasoning over long range dependencies, like the *Long Range Arena* (Tay et al., 2021).

Internally, S4 keeps a state-space based representation. Due to the way its weights are initialized, it is able to approximately "memorize" the input sequence, removing the need for an attention mechanism. Indeed, the results from Gu et al. (2022) show that the self-attention layers can be replaced by S4 layers without losing accuracy, and that it is able to effectively model long-range dependencies in data. Moreover, one of the key advantages of the S4 kernel is that its forward step can be formulated both as a convolution and as a recurrence formula, allowing fast implementation during training, when the convolution method is used, while the recurrence formula is used to generate the output step by step during inference.

S4's competitive performance in Language Modeling (LM) promises an alternative to the Transformer for other sequence modeling tasks, such as Machine Translation (MT). In this work, we explore S4-based architectures for MT. Our goal is to find the best performing S4 architecture, and we study the impact of several architectural choices on translation accuracy, namely the effect of model depth, the number of S4 blocks, and the importance of the encoder. Despite our best efforts, our top performing attention-free S4 model lags significantly (~ 4 BLEU points) behind the Transformer, with the gap increasing with input length. We hypothesize this is due to the fact that S4 compresses the source sentence to a fixed-size representation, and thus lacks a way to access the token-level states

[†]Equal contribution.

[*]Work done during an internship at Apple.

of the source, which is important for MT. As the input length increases, it becomes increasingly hard for the model to accurately store the full source sentence in a single hidden state. In contrast, the decoder cross-attention in the Transformer acts as a retrieval mechanism, allowing to accurate retrieval of the source sentence during decoding. Armed with this observation, we enhance S4 with cross-attention, and show this is enough to close the gap to the Transformer. Finally, we combine the Transformer and S4 into an hybrid architecture that outperforms both of them.

To summarize, the main contributions of the present work are:

1. We present an in-depth study of S4 for MT.
2. We provide evidence that S4 learns *self-dependencies*, i.e. dependencies between the tokens of a single sequence, but struggles to capture *cross-dependencies*, i.e. dependencies between the tokens of two sequences, as it lacks a way to retrieve prior states.
3. We show that extending S4 with an attention mechanism allows it to more accurately capture cross-dependencies and to close the gap to the Transformer on MT.

2 Background

In this section, we provide a brief overview of S4 and Machine Translation.

2.1 Structured State Space Models

The continuous state space model (SSM) is defined by:

$$\begin{aligned} x'(t) &= \boldsymbol{A}x(t) + \boldsymbol{B}u(t) \\ y(t) &= \boldsymbol{C}x(t) + \boldsymbol{D}u(t), \end{aligned} \tag{1}$$

where $u(t)$ is a 1D input signal that is mapped to the latent state $x(t)$ and finally to the output $y(t)$. $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$, and $\boldsymbol{D}$ are learned parameters. Similar to Gu et al. (2022), we assume $\boldsymbol{D} = 0$ since it is equivalent to a residual connection.

Discretization Following Gu et al. (2022), we discretize Equation (1) to apply it to discrete sequences:

$$\begin{aligned} x_k &= \overline{\boldsymbol{A}}x_{k-1} + \overline{\boldsymbol{B}}u_k \\ y_k &= \overline{\boldsymbol{C}}x_k, \end{aligned} \tag{2}$$

where $\overline{\boldsymbol{A}} \in \mathbb{R}^{N\times N}$, $\overline{\boldsymbol{B}} \in \mathbb{R}^{N\times 1}$, $\overline{\boldsymbol{C}} \in \mathbb{R}^{1\times N}$ are computed using a bilinear approximation with step size Δ[1]:

$$\begin{aligned} \overline{\boldsymbol{A}} &= (\boldsymbol{I} - \Delta/2 \cdot \boldsymbol{A})^{-1}(\boldsymbol{I} + \Delta/2 \cdot \boldsymbol{A}) \\ \overline{\boldsymbol{B}} &= (\boldsymbol{I} - \Delta/2 \cdot \boldsymbol{A})^{-1}\Delta\boldsymbol{B} \\ \overline{\boldsymbol{C}} &= \boldsymbol{C}, \end{aligned} \tag{3}$$

and $u(t)$ is sampled at $u_k = u(k\Delta)$.

Equation (2) is designed to handle 1D input signals. In practice, inputs are rarely 1D, but rather high-dimensional feature vectors, such as embeddings. To handle multiple features, Gu et al. (2022) use one independent SSM per dimension. These independent SSMs are then concatenated and mixed using a linear layer. For example, if a model has a state size of 64 and a hidden size of 512, it will contain 512 independent SSMs (Equation (1)). Each of these SSMs has a size of 64 and processes a single feature. The 1D outputs of these 512 models are concatenated, and a linear transformation is applied. This process is referred to as an *S4 block*, which involves concatenating all the independent SSMs (one Equation (2) for each feature), followed by a mixing layer, a residual connection, and Layer Normalization (Ba et al., 2016).

HiPPO Matrix A careful initialization of the $\boldsymbol{A}$ matrix is necessary to reduce exploding/vanishing gradient (Gu et al., 2022). Gu et al. (2020) proposed HiPPO-LegS matrices, which allow the state $x(t)$ to memorize the history of the input $u(t)$:

$$A_{nk} = -\begin{cases} (2n+1)^{1/2}(2k+1)^{1/2} & \text{if } n > k \\ n+1 & \text{if } n = k \\ 0 & \text{if } n < k \end{cases}$$

where A_{nk} is the entry on row n and column k. Following Gu et al. (2022), we initalize $\boldsymbol{A}$ with the above equation but train it freely afterwards.

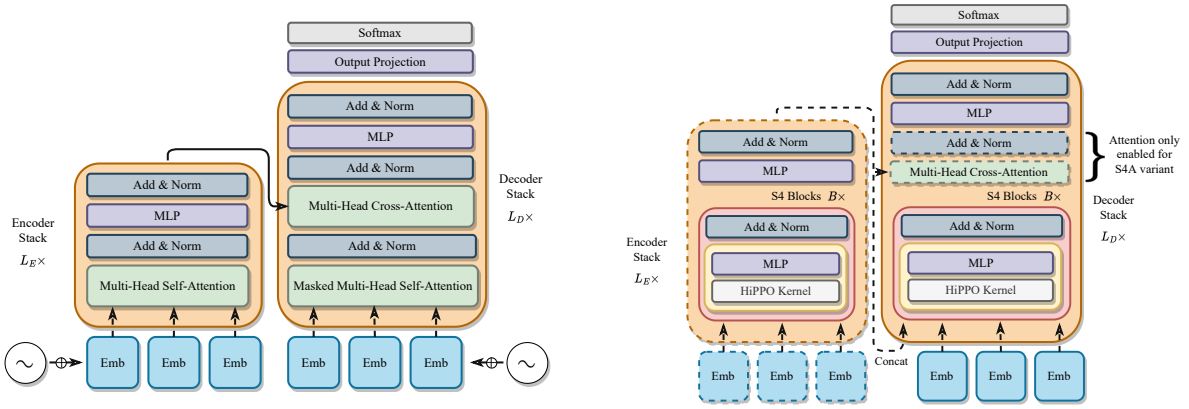
Structured State Spaces (S4) Finally, Gu et al. (2022) introduced a set of techniques to make the training of the above architecture more efficient. These include directly computing the output sequence at training time using a single convolution (denoted with $*$):

$$y = \overline{\boldsymbol{K}} * u_k. \tag{4}$$

where $\overline{\boldsymbol{K}}$ is a kernel given by:

$$\begin{aligned} \overline{\boldsymbol{K}} &:= \left(\overline{\boldsymbol{C}}\overline{\boldsymbol{A}}^i\overline{\boldsymbol{B}}\right)_{i\in[L]} \\ &= \left(\overline{\boldsymbol{C}}\overline{\boldsymbol{B}}, \overline{\boldsymbol{C}}\overline{\boldsymbol{A}}\overline{\boldsymbol{B}}, \ldots, \overline{\boldsymbol{C}}\overline{\boldsymbol{A}}^{L-1}\overline{\boldsymbol{B}}\right), \end{aligned} \tag{5}$$

[1] Since for Machine Translation the step size does not change, we use $\Delta = 1$.



(a) Transformer (TR-TR)

(b) State Spaces (S4-S4 and ∅-S4)

Figure 1: Overview of the architectures used. The Transformer architecture (a) is compared to a S4 architecture with an optional encoder (b). "Add & Norm" represents the residual connection and normalization blocks used. The attention module is used only for the S4A variant (see Section 4.4).

and L is the sequence length. At inference time, Equation (2) is applied step-by-step. For more details, see Gu et al. (2022).

2.2 Machine Translation (MT)

Let $(x_{1:n}, y_{1:m})$ be a source and target sentence pair. The negative log-likelihood of y given x can be written as:

$$-\log p(y_{1:m} \mid x_{1:n}) = -\sum_{i=1}^{m} \log p(y_i \mid x_{1:n}, y_{<i}), \quad (6)$$

where $p(y_i \mid x_{1:n}, y_{<i})$ is modeled using a neural network. In encoder-decoder models, such as the Transformer (Vaswani et al., 2017), the model has two main components: an encoder, responsible for capturing source-side dependencies, and a decoder, which captures both target-side and source-target dependencies.

Alternatively, MT can be treated as a Language Modeling task, where the (decoder-only) model is trained on the concatenated source and target sentences, separated with a special `[SEP]` token in between (Wang et al., 2021; Gao et al., 2022). Following this approach, the negative log-likelihood is written as:

$$-\log p(y_{1:m}, x_{1:n}) = \overbrace{-\sum_{j=1}^{n} \log p(x_j \mid x_{<j})}^{\mathcal{L}^{AE}} + \underbrace{-\sum_{i=1}^{m} \log p(y_i \mid x_{1:n}, y_{<i})}_{\mathcal{L}^{MT}}. \quad (7)$$

The $\mathcal{L}^{AE}$ term corresponds to the source reconstruction loss, while $\mathcal{L}^{MT}$ is identical to Equation (6). Since our focus is on MT, we only need to optimize the second term, i.e., $\mathcal{L}^{MT}$. In our experiments, including both loss terms degraded translation quality (see Appendix A). Therefore, for our decoder-only models using only the second term, $\mathcal{L}^{MT}$.

2.3 Transformer

Transformers (Vaswani et al., 2017) are the state-of-the-art architecture for MT. We show a typical architecture in Figure 1a. In particular, both encoder and decoder layers have self-attention and multi-layer perceptron (MLP) modules, and the decoder layer has an extra cross-attention module.

To simplify the text, we will refer to the architectures we discuss as [ENC]-[DEC], where [ENC] and [DEC] refer to the architecture used. For example, the Transformer model in Figure 1a will be referred to as TR-TR, since both the encoder and decoder are from the Transformer.

3 S4 for Machine Translation

3.1 Base Architecture

Following Gu et al. (2022), our architectures are based on the Transformer, but with the S4 block (Section 2) replacing self-attention. In our initial experiments, we intentionally omitted the use of cross-attention in our models to determine whether S4's internal states alone suffice in capturing long-range dependencies for MT. We call the B consecutive S4 blocks together with the MLP layer, followed by a residual connection and normalization, one *S4 layer*. Gu et al. (2022) use $B = 2$.

We consider two approaches (Figure 1b): a decoder-only model ($\emptyset-$S4), and an encoder-decoder architecture (S4-S4). Our decoder-only model is based on Gu et al. (2022), which was shown to perform well in language modeling. This model is designed to predict the next target token by taking as input the concatenated source and the previously predicted target tokens. Our S4-S4 encoder-decoder architecture consists of L_E S4 encoder layers and L_D S4 decoder layers, *without* cross-attention. Instead, we use a simple method to propagate information between the encoder and the decoder: concatenating the encoder outputs with the shifted target sequence. This way, the decoder processes both the encoder outputs and the target tokens.[2]

Finally, for some of the latter experiments, we consider the case where encoder is bidirectional, which we will refer to as S4BI. In this configuration, the S4 blocks have two sets of parameters ($\overline{\boldsymbol{A}}$, $\overline{\boldsymbol{B}}$ and $\overline{\boldsymbol{C}}$), one per direction.

3.2 S4 with Cross-Attention

In our later experiments, we employ a modified S4 decoder architecture, S4A (S4 with Attention). S4A can be used with either a Transformer or S4 encoder. It incorporates a multihead cross-attention module on top of the HiPPO kernel, as shown in Figure 1b. Specifically, cross-attention is inserted above the "Add & Norm" layer in the S4 block, followed by another "Add & Norm" layer, similar to the Transformer architecture. When cross-attention is employed, we no longer concatenate the encoder outputs to the shifted target sequence.

4 Results

In this section, we describe the experimental setup, and discuss our results.

4.1 Experimental Setup

Data We run experiments on WMT'14 English↔German (EN↔DE, 4.5M sentence pairs), and WMT'16 English↔Romanian (EN↔RO, 610K sentence pairs), allowing us to measure performance on four translation directions. For our analysis, we focus on EN→DE. We tokenize all data using the Moses tokenizer and apply the Moses scripts (Koehn et al., 2007) for punctuation normalization. We use Byte-pair encoding (BPE, Sennrich et al. (2016)) with $40,000$ merge operations, and the WMT'16 provided scripts to normalize EN↔RO for the RO side, and to remove diacritics when translating RO→EN. Translations into Romanian keep diacritics to generate accurate translations. We evaluate using `sacreBLEU`[3] version `2.1.0` (Post, 2018), with signature `nrefs:1 | case:mixed | eff:no | tok:13a | smooth:exp`. We run all experiments using FAIRSEQ (Ott et al., 2019), onto which we ported the code from Gu et al. (2022)[4].

Unless stated otherwise, we report BLEU scores on the WMT'14 EN→DE validation set.

Hyperparameters We optimize using ADAM (Kingma and Ba, 2015). After careful tuning, we found the best results with a learning rate of 0.005 for the S4 models, 0.001 for the Transformer models, and 0.002 for the hybrid models. We train for 100 epochs (28 000 steps), by which point our models had converged, and average the last 10 checkpoints. We use $4\,000$ warm-up steps and an inverse square root learning rate scheduler (Vaswani et al., 2017). We used a dropout rate of 0.1 for EN↔DE, and 0.3 for EN↔RO. Unless stated otherwise, all models have layer and embedding sizes of 512, the hidden size of the feed-forward layers is 2048, and we use 8 attention heads for the Transformer. For both the Transformer and S4, we use post-normalization[5]. Following Gu et al. (2022) we use GeLU activation (Hendrycks and Gimpel, 2016) after the S4 modules and GLU activation (Dauphin et al., 2017) after the linear layer.

S4-specific Training Details During our exploration, we experimented with several choices that had a marginal effect on performance:

(i) *Module-specific learning rates.* Gu et al. (2022) suggested different learning rates for the matrices in eq. (2) and the neural layer, but we did not observe any significant difference.

(ii) *Trainable $\overline{\boldsymbol{A}}$ and $\overline{\boldsymbol{B}}$.* In line with Gu et al. (2022), freezing $\boldsymbol{A}$ and $\boldsymbol{B}$ did not cause a noticeable performance drop.

(iii) *State dimension.* We varied the size of the state (x_k in Equation (2)), but found that that

[2] Ideally, we would initialize the S4 decoder state spaces with the last state of the encoder. However, this is non-trivial to implement, since the forward step is executed as a single convolution during training. We leave the exploration of this method to future work.

[3] `https://github.com/mjpost/sacrebleu`

[4] `https://github.com/HazyResearch/state-spaces`

[5] In our experiments, we didn't observe any difference between pre and post-normalization.

increasing it dimension beyond 64 did not noticeably affect translation quality. Therefore, similarly to Gu et al. (2022), we set the state dimension to 64 in our experiments. Note that this parameter should not be confused with the model's hidden size, which we examine in Section 4.2. Increasing the state dimension increases the modeling capacity of the S4 kernel for **each input** dimension, but the output is still collapsed to the hidden size, making the latter the bottleneck.

(iv) *Learning rate scheduler.* We observed no significant difference between using the inverse square root scheduler and the cosine scheduler suggested in (Gu et al., 2022).

4.2 Parameter Allocation and Scaling

Encoder Scaling To explore the effect of parameter allocation on performance, we compare the translation quality of different encoder-decoder configurations with the same total number of parameters (roughly 65M). In Figure 2a, the x axis represents the ratio of encoder layers to the total number of layers (encoder + decoder). Starting with a decoder-only model (ratio $= 0$), we gradually increase the number of encoder layers, and end with a model containing only a single decoder layer. Two results stand out: first, there is a wide gap between the best S4 and Transformer models: 20.7 and 26.4 BLEU, respectively. Second, and consistent with prior work, we find that an even split of parameters between the encoder and decoder (6 encoder layers and 6 decoder layers, i.e., Transformer base) yields the best translation quality for the Transformer (Vaswani et al., 2017), whereas no encoder produces the best results for S4. Based on this finding, we focus on the S4 decoder-only variant for the next experiments.

Number of S4 Blocks per Layer Prior research set the number of S4 blocks, B, to 2 (Gu et al., 2022). We found that increasing B is beneficial as S4 blocks are responsible for capturing dependencies between tokens. In Table 1 we vary B while keeping the parameter count roughly constant. Increasing B leads to noticeable quality improvements until $B = 10$. This architecture achieves a score of 22.7 BLEU, but the gap to the Transformer is still substantial: 3.7 BLEU points. From here onward we use $B = 10$ and 6 layers for the decoder-only model, unless stated otherwise.

B	L_D	$\lvert\boldsymbol{\theta}_{\text{S4}}\rvert$	$\lvert\boldsymbol{\theta}\rvert$	BLEU
1	17	10M	66M	20.0
2	14	20M	66M	20.7
3	12	21M	66M	21.2
4	10	23M	64M	21.5
6	8	28M	64M	22.1
10	6	35M	67M	**22.7**
16	4	37M	65M	22.0
22	3	38M	64M	22.2
35	2	40M	64M	22.5

Table 1: Effect of number of S4 blocks per layer on the decoder-only architecture. B is the number of S4 blocks, L_D the number of decoder layers, $\lvert\boldsymbol{\theta}_{\text{S4}}\rvert$ are the parameters allocated for S4 inside the HiPPO kernels, and $\lvert\boldsymbol{\theta}\rvert$ are the total parameters.

	Short $[1, 17]$	Medium $[18, 29]$	Long $[30, 117]$	Overall
TR-TR	25.9	26.8	26.4	26.4
S4-Normal	24.0	24.3	21.4	22.7
S4-Reverse	23.2	24.2	22.5	23.1

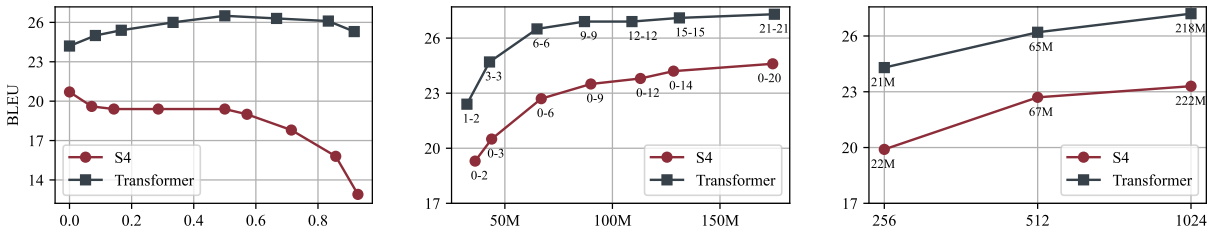
Table 2: Translation quality of S4, trained on regular and reversed source sentences, compared to Transformer on the WMT'14 EN-DE validation set, for different reference sentence lengths. Each bucket has approximately 1k sentences.

Depth Scaling In Figure 2b we show BLEU as we increase the number of layers. The x axis shows the total number of parameters of each architecture, and the numbers next to each data point indicate the architecture (e.g., 1-2 means a 1 layer encoder and 2 layer decoder). There is a clear gap in performance between the two models, which is decreasing as more layers are added, i.e. S4 seems to benefit more from increasing the number of layers.

Width Scaling In Figure 2c we examine the influence of the hidden size on both S4 and Transformer, for the 0-6 and 6-6 architectures, respectively. While S4's performance improves with increasing width, the returns are diminishing, and the gap to the Transformer does not go way.

4.3 Translation Quality Comparison

Despite our extensive tuning of the S4 architecture, a gap of almost 4 BLEU points to the Transformer remains. In this section, we delve deeper into S4's results to determine why it is struggling.



(a) Encoder parameter allocation (*ratio*). (b) Number of parameters (*depth*). (c) Hidden size (*width*).

Figure 2: Scaling plots for S4 and the Transformer. We explore shifting the parameter allocation between the encoder (a), depth scaling (with a fixed hidden size of 512), symmetrically for the encoder-decoder Transformer, and on the decoder for S4 (b), and hidden size (width) scaling (c), with 0-6 and 6-6 layers of S4 and Transformer, respectively.

Sentence Length In Table 2, we split the source sentences into 3 buckets according to their length[6], and show the BLEU scores for both S4 and the Transformer. There is a clear gap between the two models, which increases with sentence length. Specifically, the gap is 1.9 and 2.5 BLEU for short and medium-length sentences, respectively, but it increases to 5 for the longest bucket. This observation is not entirely surprising: S4 uses a fixed-size vector to compress the full source sentence and the previous target tokens, which is not enough for long sentences. The Transformer, on the other hand, has no such constraint, as its attention mechanism lets it retrieve previous states as needed.

Reversing Source Sentences To further investigate whether the limited representation size is causing the poor performance of the model, we applied a technique from the earlier neural MT literature. Before the introduction of attention (Bahdanau et al., 2015), it was observed that reversing the source sequence could improve performance by decreasing the distance between cross-language dependencies (Sutskever et al., 2014). We trained a model on reversed source sentences, and report the results in Table 2 as S4-Reverse. Compared with the regular model, we get a small overall improvement of 0.4 BLEU points, but a large improvement of 1.1 BLEU on long sentences. This observation suggests that although the HiPPO matrix has promising temporal characteristics, S4 is not able to adequately represent the source sentence and utilize its content during the decoding phase.

[6]To limit spuriousness issues, we chose the buckets so that each bucket has roughly 1k sentences.

4.4 The Importance of Attention

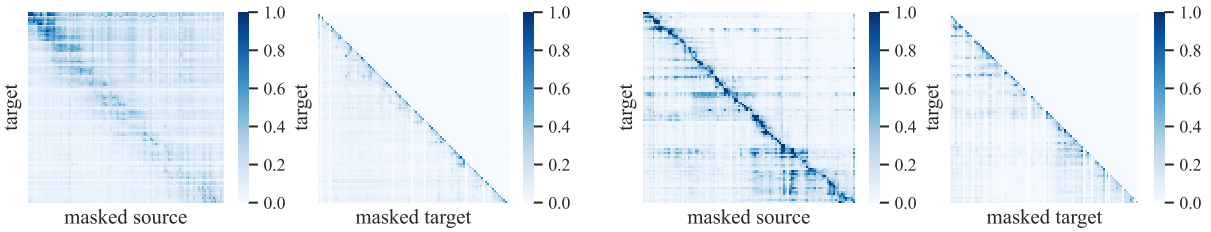
In the previous section, we showed that S4 struggles to translate long sentences. In this section, we study the influence of each source token on the output of the model.

Attention Heatmaps To investigate the extent to which S4 captures dependencies between source and target words, we use a method from He et al. (2019). For each generated target token, we mask out the source tokens, one by one, and replace them with padding tokens. Then, we measure the relative change in the decoder's final layer activation caused by this intervention using L2 distance. By repeating this process for each source token, we obtain a two-dimensional matrix measuring the impact of each source token on each target token. Similarly, we can perform the same procedure by masking the previous target tokens to obtain a similar plot for target-side self-dependencies.

We show the heatmaps for both S4 and the Transformer[7] in Figure 3. As shown, the differences are stark. The Transformer is focused on just a few words (sharp diagonal in fig. 3b), while S4 is is much more "blurred" and unable to appropriately attend to specific parts of the source sentence. The difference is not as pronounced for short sentences (see Figure 4), indicating that a single hidden state is not enough to capture all the information the model needs for longer sentences.

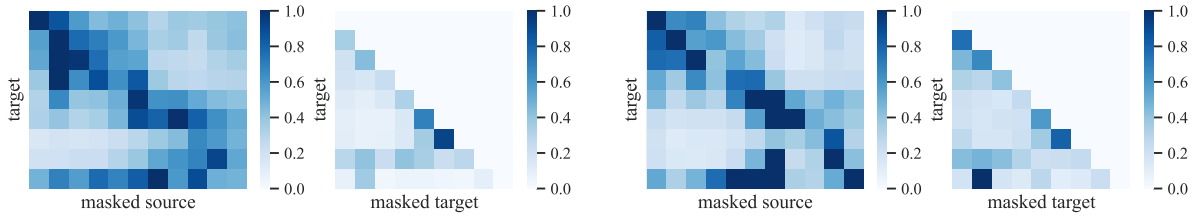
In Appendix B, we explore how B impacts the heatmaps. We find that increasing B sharpens the heatmaps, although they never get as sharp as those of the Transformer.

[7]The plots are qualitatively similar to the usual attention weights heatmaps for the Transformer. We show these "masking" maps for both models for fair comparison.



(a) ∅-S4 (b) TR-TR

Figure 3: Change in the final decoder hidden state for each generated token when masking out source and target tokens in one *long* sample of EN-DE (109 tokens), for the decoder-only S4 (a) and the Transformer (b). While the latter can discriminate between source words very accurately (sharp diagonal in b), S4 fails to do so.



(a) ∅-S4 (b) TR-TR

Figure 4: Change in the final decoder hidden state for each generated token when masking out source and target tokens in one *short* sample of EN-DE (11 tokens) for the decoder-only S4 and the Transformer. In the case of short sentences, S4 is able to more accurately align source and target words.

4.5 Attention-enhanced Architectures

In the previous experiments, we found that S4 underperforms on long sentences, and hypothesized that this is due to its fixed-size representation, which makes it unable to recall the full source sentence. To address this, we now extend the S4 decoder with an attention mechanism, which allows us to use an encoder-decoder setup, S4-S4A. For more details on the attention mechanism, see Section 3.2.

We conducted experiments similar to those in Section 4.2 to determine the optimal B and how to allocate layers to the encoder and the decoder, while keeping the total number of parameters constant. We summarize the findings in Tables 3 and 4. We found the best results with a balanced architecture, $5-5$, and $B=3$. This model improves performance by almost 3 BLEU points on the WMT'14 validation set, from 22.7 to 25.6. From here onward, encoders and decoders have 5 layers for S4 and 6 layers for Transformer.

In Table 5 we compare the performance of S4-S4A and the Transformer (TR-TR) for short, medium, and long sentences. Although there is a noticeable improvement over the attention-free S4 model (∅-S4), especially for longer sentences, there is still gap between the two models. One possible explanation for the comparatively poorer performance of S4-S4A is the unidirectional nature of the S4 encoder. This results in subpar representations for the initial words in the source sentence. Indeed, when using a S4 encoder with a Transformer de-

B	L_E	L_D	$\lvert\boldsymbol{\theta}\rvert$	BLEU
2	6	6	66M	24.9
3	5	5	64M	**25.4**
5	4	4	64M	**25.4**
8	3	3	63M	25.2

Table 3: Effect of number of B and number of encoder (L_E) and decoder (L_D) layers for the S4-S4A encoder-decoder architecture.

L_E	1	2	3	4	5	6	7	8	9
L_D	9	8	7	6	5	4	3	2	1
BLEU	24.5	24.8	25.1	25.1	**25.4**	25.1	25.1	25.1	23.7

Table 4: Effect of allocating layers to the encoder or to the decoder on the S4-S4A architecture, with $B=3$. The models have a total of 10 layers between the encoder and decoder.

	Short [1, 17]	Medium [18, 29]	Long [30, 117]	Overall
∅-S4	24.0	24.3	21.4	22.7
TR-TR	**25.9**	26.8	26.4	26.4
S4-TR	24.7	25.5	25.2	25.2
S4-S4A	25.0	26.5	25.3	25.6
S4BI-TR	25.5	25.9	25.6	25.7
S4BI-S4A	25.3	26.5	25.8	25.9
TR-S4	24.2	24.8	22.9	23.7
TR-S4A	25.6	**26.9**	**26.5**	**26.5**

Table 5: Translation quality of different attention-enhanced models on the WMT'14 EN-DE validation set for different source sentence lengths. Each bucket has approximately 1k sentences. All models have $64M < |\boldsymbol{\theta}| < 66M$ parameters.

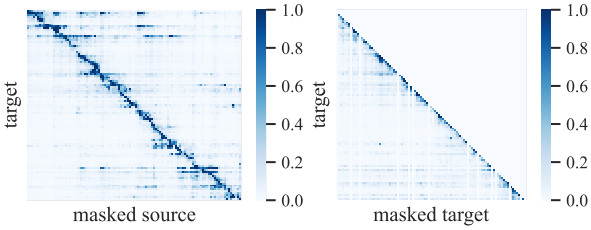


Figure 5: Comparison of TR-S4A's change in the final decoder hidden state for each generated token when masking out source tokens for one *long* sample of EN-DE (the same sample as Figure 3). Enhancing S4 with attention helps it to focus on the source tokens, similar to TR-TR.

coder (S4-TR), the performance is still behind that of TR-TR, and replacing the S4 encoder with a Transformer (TR-S4A) allows us to match the performance Transformer. Making the S4 encoder bidirectional (S4BI), we are able to narrow the performance gap to the Transformer to just 0.5 BLEU points (see S4BI-S4A).

Finally, in Figure 5 we show the attention heatmaps for TR-S4A architecture, which were generated in the same was as those in Figure 3. These plots show that the model is now capable of accurately aligning source and target words, and are qualitatively similar to those of the Transformer.

Why does S4 perform well on LM but not MT? A natural question to ask is why does S4 perform well on LM (Gu et al., 2022), but not on MT. Our intuition is that MT is a more challenging task. For LM, the model only needs to consider a shorter context to accurately predict the next token, whereas

	EN-DE	DE-EN	EN-RO	RO-EN
∅-S4	22.1	25.4	12.8	19.7
S4BI-S4A	26.1	29.5	22.7	31.0
TR-S4A	**27.3**†	**31.4**	**24.1**†	**33.6**†
TR-TR	26.9	**31.4**	23.8	33.2

Table 6: BLEU scores on test set for each architecture in 4 different language pairs. The † on TR-S4A indicates statistically significant results.

for MT, it requires accurate access to the source sentence representations. As the length of the source sentence increases, a fixed-size state is insufficient to capture fine-grained representations of the source, and thus the model's performance suffers. This is in line with the observations made by Vig and Belinkov (2019), who argue that Transformer LMs tend to pay more attention to the previous few tokens, emphasizing the importance of short-term memory over long-term memory.

4.6 Results for Other Language Pairs

In the previous sections, we focused on EN-DE. In this section, we compare the different S4 architectures for other language pairs (DE-EN, EN-RO, and RO-EN) and summarize the results in Table 6. These numbers are on the test sets of the respective language pairs. The results align with our previous findings. Without attention, there is a significant gap between S4 and the Transformer models, which is reduced significantly by adding it. Interestingly, the best performing architecture for all language pairs is the hybrid TR-S4A, which provides a small but statistically significant[8] improvement over the Transformer for all but DE→EN.

5 Conclusion and Future Work

In this work, we explored the application of S4 to Machine Translation and conducted an investigation into the best architecture and hyperparameters. Despite our efforts, we found that S4's translation accuracy lagged behind the Transformer, and the performance gap widened for longer sentences. We then showed that this was due to the limitations of the fixed-size representation used by S4, which had to compress the entire prior context, including the source sentence and previous output tokens. Finally, we showed that the performance gap can be closed by incorporating attention.

[8]We performed statistical significance tests using paired bootstrap resampling (Koehn, 2004) and a significance of 5%.

Since we did our investigation into S4, numerous new SSM models have been proposed. Of particular note are S5 (Smith et al., 2023), which utilizes a multi-input multi-output SSM, instead of one single-input single-output SSM per feature as S4 does, and H3 (Dao et al., 2023), which is faster and better at LM than S4. We hope future research explores how well these models perform on MT. Additionally, it is worth noting MEGA (Ma et al., 2023), which incorporates SSM's into the Transformer attention, and is effective in MT, albeit at the expense of quadratic complexity.

6 Acknowledgements

We would like to thank António V. Lopes, Hendra Setiawan, and Matthias Sperber for their suggestions and feedback. Their contributions significantly improved the final work.

References

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

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

Brown, Tom B., Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In Larochelle, Hugo, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Dao, Tri, Daniel Y. Fu, Khaled K. Saab, Armin W. Thomas, Atri Rudra, and Christopher Ré. 2023. Hungry Hungry Hippos: Towards language modeling with state space models. In *International Conference on Learning Representations*.

Dauphin, Yann N., Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In Precup, Doina and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 933–941. PMLR.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June. Association for Computational Linguistics.

Gao, Yingbo, Christian Herold, Zijian Yang, and Hermann Ney. 2022. Is encoder-decoder redundant for neural machine translation? In He, Yulan, Heng Ji, Yang Liu, Sujian Li, Chia-Hui Chang, Soujanya Poria, Chenghua Lin, Wray L. Buntine, Maria Liakata, Hanqi Yan, Zonghan Yan, Sebastian Ruder, Xiaojun Wan, Miguel Arana-Catania, Zhongyu Wei, Hen-Hsen Huang, Jheng-Long Wu, Min-Yuh Day, Pengfei Liu, and Ruifeng Xu, editors, *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing, AACL/IJCNLP 2022 - Volume 1: Long Papers, Online Only, November 20-23, 2022*, pages 562–574. Association for Computational Linguistics.

Goel, Karan, Albert Gu, Chris Donahue, and Christopher Ré. 2022. It's raw! audio generation with state-space models. In Chaudhuri, Kamalika, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 7616–7633. PMLR.

Gu, Albert, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. 2020. Hippo: Recurrent memory with optimal polynomial projections. In Larochelle, Hugo, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Gu, Albert, Karan Goel, and Christopher Re. 2022. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*.

He, Shilin, Zhaopeng Tu, Xing Wang, Longyue Wang, Michael Lyu, and Shuming Shi. 2019. Towards understanding neural machine translation with word importance. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 953–962, Hong Kong, China, November. Association for Computational Linguistics.

Hendrycks, Dan and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

Kingma, Diederik P. and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In Bengio, Yoshua and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic, June. Association for Computational Linguistics.

Koehn, Philipp. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 388–395, Barcelona, Spain, July. Association for Computational Linguistics.

Ma, Xuezhe, Chunting Zhou, Xiang Kong, Junxian He, Liangke Gui, Graham Neubig, Jonathan May, and Luke Zettlemoyer. 2023. Mega: Moving average equipped gated attention. In *The Eleventh International Conference on Learning Representations*.

NLLB Team, Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loic Barrault, Gabriel Mejia Gonzalez, Prangthip Hansanti, John Hoffman, Semarley Jarrett, Kaushik Ram Sadagopan, Dirk Rowe, Shannon Spruit, Chau Tran, Pierre Andrews, Necip Fazil Ayan, Shruti Bhosale, Sergey Edunov, Angela Fan, Cynthia Gao, Vedanuj Goswami, Francisco Guzmán, Philipp Koehn, Alexandre Mourachko, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, and Jeff Wang. 2022. No language left behind: Scaling human-centered machine translation. *arXiv preprint arXiv:2207.04672*.

Ott, Myle, 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 *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota, June. Association for Computational Linguistics.

Post, Matt. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium, October. Association for Computational Linguistics.

Sennrich, Rico, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August. Association for Computational Linguistics.

Smith, Jimmy T.H., Andrew Warrington, and Scott Linderman. 2023. Simplified state space layers for sequence modeling. In *The Eleventh International Conference on Learning Representations*.

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In Ghahramani, Zoubin, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Tay, Yi, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. 2021. Long range arena : A benchmark for efficient transformers. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, pages 1–19.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Guyon, Isabelle, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Vig, Jesse and Yonatan Belinkov. 2019. Analyzing the structure of attention in a transformer language model. In Linzen, Tal, Grzegorz Chrupala, Yonatan Belinkov, and Dieuwke Hupkes, editors, *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@ACL 2019, Florence, Italy, August 1, 2019*, pages 63–76. Association for Computational Linguistics.

Wang, Shuo, Zhaopeng Tu, Zhixing Tan, Wenxuan Wang, Maosong Sun, and Yang Liu. 2021. Language models are good translators. *arXiv preprint arXiv:2106.13627*.

A Influence of $\mathcal{L}^{AE}$

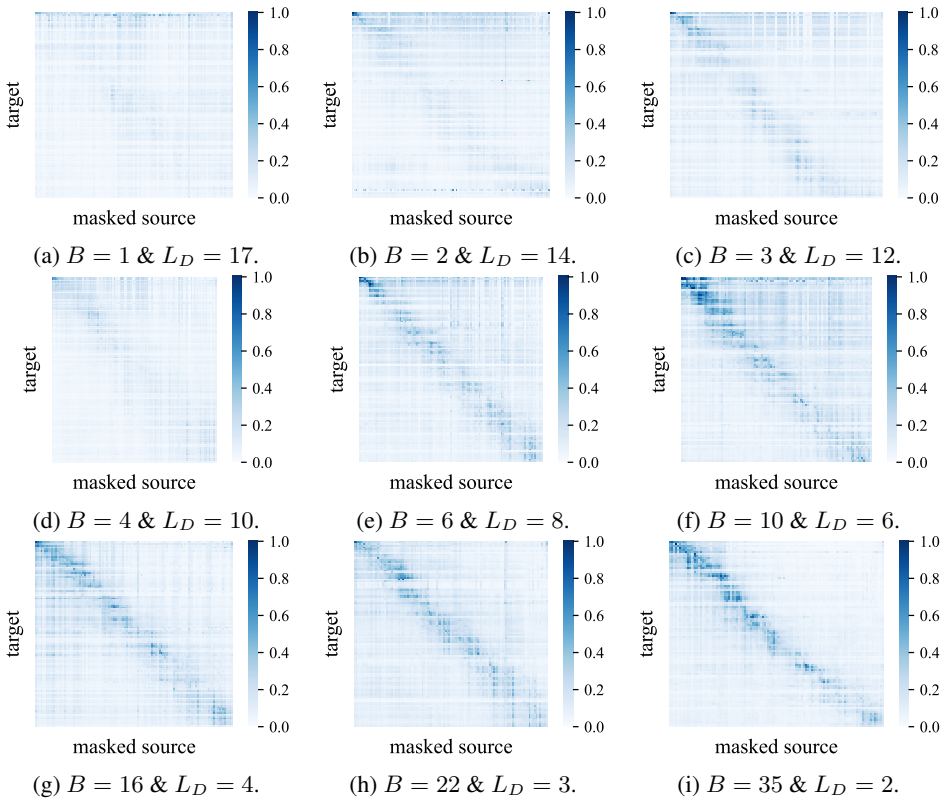
In our experiments with the decoder-only architecture, we intentionally excluded the loss term $\mathcal{L}^{AE}$ from Equation (6) as it is not necessary for MT. In Table 7 we show the effect of including this loss during training: performance degradation of around 4 BLEU points for both architectures.

B	L_D	$\lvert\boldsymbol{\theta}\rvert$	w/ $\mathcal{L}^{AE}$	w/o $\mathcal{L}^{AE}$
6	8	65M	17.9	22.3
10	6	68M	18.6	22.5

Table 7: Impact of the autoencoder loss ($\mathcal{L}^{AE}$) on translation quality on the WMT'14 validation set for two decoder-only architectures. B is the number of S4 blocks, L_D the number of decoder layers (this is a decoder-only architecture), and $|\boldsymbol{\theta}|$ is the number of parameters.

B Effect of B in the Cross-Attention Heatmaps

Using the methodology described in Section 4.4, Figure 6 shows the cross-attention heatmaps for the models in Table 1. All models have roughly the same number of parameters, and differ only in B and the number of layers (L_D). As in Figure 3, the source sentence has 109 tokens. A noticeable pattern emerges: as B increases, the heatmap sharpens, meaning it is easier for S4 to retrieve the source states. It is worth noting, however, that these heatmaps never get as sharp as those of the models with attention.



(a) $B = 1$ & $L_D = 17$. (b) $B = 2$ & $L_D = 14$. (c) $B = 3$ & $L_D = 12$.

(d) $B = 4$ & $L_D = 10$. (e) $B = 6$ & $L_D = 8$. (f) $B = 10$ & $L_D = 6$.

(g) $B = 16$ & $L_D = 4$. (h) $B = 22$ & $L_D = 3$. (i) $B = 35$ & $L_D = 2$.

Figure 6: Cross-attention heatmaps for the models in Table 1. Increasing B (while keeping the total number of parameters roughly constant) makes the heatmaps less blurry, which means it is easier for the model to retrieve source states.