Journal of Artificial Intelligence Research 75 (2022) 677-708 Submitted 05/2022; published 10/2022

AAN+: Generalized Average Attention Network for Accelerating Neural Transformer

Biao Zhang ZB@STU.XMU.EDU.CN
School of Informatics, Xiamen University, Xiamen 361005, China
School of Informatics, University of Edinburgh
Edinburgh EH8 9AB, United Kingdom

Deyi Xiong DYXIONG@TJU.EDU.CN
College of Intelligence and Computing, Tianjin University
Tianjin 300350, China

Yubin Ge YUBINGE2@ILLINOIS.EDU
University of Illinois at Urbana-Champaign, Urbana, IL61801, USA

Junfeng Yao YAO0010@XMU.EDU.CN
School of Film, Xiamen University, Xiamen 361005, China

Hao Yue YHAO1022@163.COM
School of Informatics, Xiamen University, Xiamen 361005, China

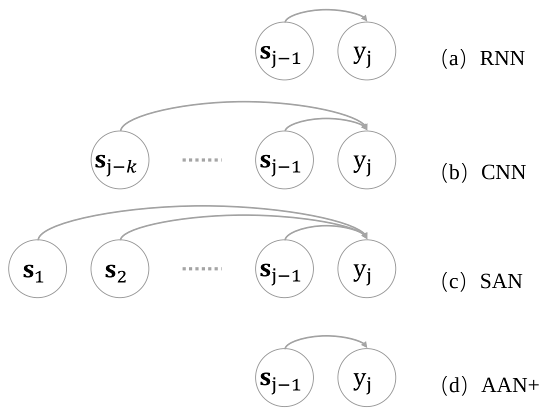
Jinsong Su (*corresponding author*) JSSU@XMU.EDU.CN
School of Informatics, Xiamen University
Key Laboratory of Digital Protection and Intelligent Processing of Intangible Cultural Heritage of Fujian and Taiwan, Ministry of Culture and Tourism
Xiamen 361005, China

Abstract

Transformer benefits from the high parallelization of attention networks in fast training, but it still suffers from slow decoding partially due to the linear dependency $\mathcal{O}(m)$ of the decoder self-attention on previous target words at inference. In this paper, we propose a generalized average attention network (AAN+) aiming at speeding up decoding by reducing the dependency from $\mathcal{O}(m)$ to $\mathcal{O}(1)$. We find that the learned self-attention weights in the decoder follow some patterns which can be approximated via a dynamic structure. Based on this insight, we develop AAN+, extending our previously proposed average attention (Zhang et al., 2018a, AAN) to support more general position- and content-based attention patterns. AAN+ only requires to maintain a small constant number of hidden states during decoding, ensuring its $\mathcal{O}(1)$ dependency. We apply AAN+ as a drop-in replacement of the decoder self-attention and conduct experiments on machine translation (with diverse language pairs), table-to-text generation and document summarization. With masking tricks and dynamic programming, AAN+ enables Transformer to decode sentences around 20% faster without largely compromising in the training speed and the generation performance. Our results further reveal the importance of the localness (neighboring words) in AAN+ and its capability in modeling long-range dependency.

1. Introduction

In recent years, Transformer (Vaswani et al., 2017) has become the dominant model in machine translation and achieved state-of-the-art results on various language pairs (Barrault



$\mathbf{s}_{j-1}$ y_j (a) RNN
$\mathbf{s}_{j-k}$ $\mathbf{s}_{j-1}$ y_j (b) CNN
$\mathbf{s}_1$ $\mathbf{s}_2$ $\mathbf{s}_{j-1}$ y_j (c) SAN
$\mathbf{s}_{j-1}$ y_j (d) AAN+

Figure 1: Illustration of the decoding procedure for different neural architectures. We show which previous target hidden states s_* are used to predict the current target word y_j. k denotes the filter size of the convolution layer. `AAN+` reduces the decoding dependency from $\mathcal{O}(m)$ (SAN) to $\mathcal{O}(1)$. Note that $\mathbf{s}_*$ might have different meanings across different models, here only used for intuitive illustration.

et al., 2019). Under the encoder-decoder framework (Bahdanau et al., 2015), it uses attention networks as its backbone to summarize source and target sentence semantics and to induce source-target translation correspondence. The attention network is highly parallelizable and capable of handling long-range dependency within a short path. These advantages ensure the superiority of Transformer in training speed and translation quality against its alternatives, including recurrent neural network (RNN) based NMT (Sutskever et al., 2014; Bahdanau et al., 2015; Luong et al., 2015; Zhang et al., 2018b) and convolutional neural network (CNN) based NMT (Gehring et al., 2017a, 2017b).

However, despite its training efficiency, Transformer suffers from slow decoding. Figure 1 illustrates the problem, contrasting the decoding procedure of SAN, RNN and CNN. To capture the dependency on previously generated target words, SAN requires to calculate adaptive attention weights with all previous hidden states in the decoder (Figure 1c), leading to a linear decoding dependency, i.e. $\mathcal{O}(m)$.[1] Instead, CNN depends on only previous k target states (Figure 1b) and RNN relies on just one (Figure 1a), both of which show a constant decoding dependency. Under the dominant autoregressive schema, the prediction of the current word must wait until all previous words are generated; thus, the model with more complex decoding dependency will inevitably suffer from worse inference efficiency. Empirically, our breakdown analysis shows that the decoder-side SAN alone takes about 41.42% of the decoding time (Zhang et al., 2019), which forms a significant inference barrier.

In this paper, we explore ways to reduce such dependency for SAN, aiming at accelerating the inference of Transformer but without degrading its performance. One pioneer study is average attention network (`AAN`) proposed in our previous conference paper (Zhang et al., 2018a), where we heuristically assign an equal weight to each previous hidden state.

1. In this paper, we use the term *decoding dependency* to denote the number of previous hidden states required for predicting the next target word in one-step translation. Notice that models with short decoding dependency, like RNN and `AAN+`, can also handle long-range dependency because distant information can be encoded into one compact hidden state.

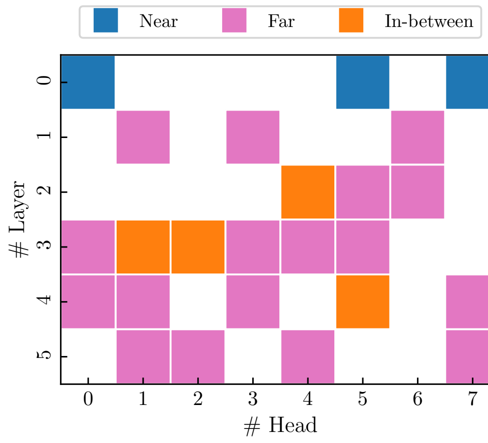


Figure 2: Position-based analysis for the learned self-attention weights in the decoder on the WMT14 En-De test set. Results are for Transformer base: 6 layers (y-axis) and 8 heads (x-axis). We mainly analyze attention weights for query words starting from position 10 to reduce noises, and focus on the attended position that has the largest attention probability. If over 90% of the attended positions correspond to neighbouring/previous 3 words (the first 3 words), we mark the attention head as *Near* (*Far*); if a head focuses on the other words in over 20% cases, we mark it as *In-between*. Heads without labels are likely input or content-dependent, and can hardly be interpreted by position alone.

However, our inspection on the learned self-attention weights in the decoder shows richer patterns beyond the simple average (Voita et al., 2019): some attention heads favor neighbouring inputs, some favor distant inputs while some might be content-dependent as in Figure 2.[2] Based on this insight, we further extend `AAN` to generalized average attention network (`AAN+`), expecting to facilitate the modeling of different attention patterns.

Specially, `AAN+` formalizes the philosophy behind `AAN` via a dynamic structure and supports both position-based and content-based patterns. During training, these patterns can be pre-calculated using masking tricks to enable high parallelization like SAN; during decoding, `AAN+` leverages dynamic programming that decomposes each pattern into a few recurrent hidden states so as to reduce the decoding dependency to $\mathcal{O}(1)$ like RNN. Figure 3 shows the high-level architecture of `AAN+`, which involves two modules: an *average attention module* and a *gating network module*. The former is used to handle intra-sentence dependency, and the latter improves the model's expressiveness in describing its inputs.

The average attention network was first presented in our previous conference paper (Zhang et al., 2018a). In this paper, we make the following significant extensions:

- We extend `AAN` to `AAN+`, an average attention family that is parallelizable in training and has a decoding dependency of $\mathcal{O}(1)$. Apart from the average pattern, `AAN+` allows us to study richer attention patterns for translation, including position-based and content-based attention patterns. We provide a theoretical analysis on the decoding acceleration achieved by this attention family.

2. In this work, we formulate the *Far* pattern based on our empirical observation: many self-attention heads in the decoder just attend to the first few words of a sentence. We used 3 for illustration.

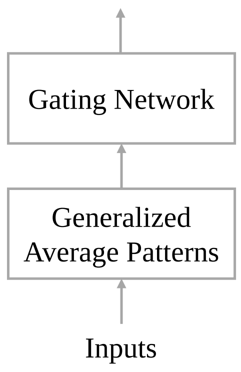


Figure 3: Visualization of the architecture of `AAN+`.

- We discover the importance of localness (neighboring words) for generation: the corresponding pattern delivers better generalization ability and comparable decoding efficiency against the vanilla `AAN` across different tasks. Also, different patterns are complementary to each other with a varying degree, whose combination could outperform SAN in terms of BLEU.
- We conduct experiments not only on machine translation following our previous paper but also additionally on table-to-text generation and document summarization. On all tasks, our model speeds up the generation by around 20% yet still achieves competitive performance against the standard Transformer baseline.

Our proposed model has been applied to several online translation services and has been adopted in Marian (Junczys-Dowmunt et al., 2018).

2. Related Work

The design of `AAN+` is inspired by RNN (Chung et al., 2014; Hochreiter & Schmidhuber, 1997). RNNs have been widely used for NMT to deal with long-range dependency (Bahdanau et al., 2015; Sutskever et al., 2014). By summarizing all previous target information into one hidden state, RNN benefits NMT with a short decoding dependency of $\mathcal{O}(1)$. Unfortunately, RNN performs sequential modeling which almost kills its parallel computation even at training. To facilitate large-scale training, researchers thereby resort to feed-forward networks. Gehring et al. (2017a) used CNN-based encoder as an alternative to RNN, and Gehring et al. (2017b) further developed a pure CNN-based NMT model with deep architectures modeling distant context. Instead of following the recurrence or the convolution, Vaswani et al. (2017) introduced Transformer, a neural architecture that relies solely on attention networks for translation. However, as we discussed in Section 1, Transformer suffers from decoding inefficiency despite its fast training speed and great performance.

The average attention family behind `AAN+` is a new type of attention network. Each hidden state in `AAN+` is a weight-sum of previous inputs, with attention weights defined by some heuristic rules. This property resonates with the recent findings on simplified RNNs (Lee et al., 2017; Zhang et al., 2018b; Lei et al., 2018; Zhang & Sennrich, 2019) and hard-coded attentions (You et al., 2020; Raganato et al., 2020). The attention network was originally proposed to capture source-target alignment relationship (Bahdanau et al., 2015). Recent efforts further improved its accuracy and capability, and extended its usage to more scenarios. Luong et al. (2015) explored several formulations for attention, distinguishing local attention from the global one. Zhang et al. (2020) proposed to update attention outputs with a GRU-gated network, which enhances the discrimination ability of context vectors for predicting the next target word. Kim et al. (2017) further studied intrinsic structures inside attention through graphical models. Shen et al. (2018) introduced a directional structure into SAN to integrate long-range dependency and temporal order information. Zhang et al. (2019) developed a merged attention network to shorten the decoder depth. Mi et al. (2016) and Liu et al. (2016) leveraged standard word alignments to supervise the automatically generated attention weights. Different from these studies, we aim to accelerate the decoding procedure by simplifying SAN. Note that the design of our model partially follows the highway network (Srivastava et al., 2015) and the residual network (He et al., 2015).

Our work falls into the category of speeding up the decoding of Transformer. In this aspect, Gu et al. (2018) proposed non-autoregressive NMT, which speeds up translation by directly generating target words without relying on previous predictions. However, their model achieves the improvement in decoding speed at the cost of worse translation quality. Our model, instead, not only achieves a remarkable gain in terms of decoding speed, but also preserves the translation performance. Wang et al. (2018), Stern et al. (2018) and Zhang et al. (2020) proposed to generate multiple target words at one time with the aim of shortening sequential decoding steps, which is orthogonal to our work. By contrast, Chelba et al. (2020) explored a window-based approach for the decoder SAN, only allowing access to previous k target words. Although this method shows superiority to the vanilla SAN, its decoding dependency relies on the window size, making it inferior to `AAN+` theoretically.

There is another line of research focusing on reducing the quadratic complexity of the (encoder) SAN for Transformer to improve its computational and memory efficiency, including Reformer (Kitaev et al., 2020), Performer (Choromanski et al., 2021), Longformer (Beltagy et al., 2020), Poolingformer (Zhang et al., 2021), and so on. Despite their success, these studies mainly improve the *encoding* of long documents and can hardly be adapted to simplify the decoding dependency. Among them, Katharopoulos et al. (2020) proposed linear attention and Peng et al. (2021) proposed random feature attention, which can decompose the attention over target words and were reported to achieve encouraging decoding speedup. Unfortunately, they both rely on a matrix-based recurrent state for decoding rather than a vector-based state, limiting their efficiency.

3. Background

In this section, we briefly review the vanilla self-attention network in Transformer (Vaswani et al., 2017, SAN), followed by an elaboration on our published pioneer study, *average attention network* (Zhang et al., 2018a, `AAN`).

3.1 Self-Attention Network

Transformer relies on SAN to handle intra-sentence dependencies by directly estimating the correlation of each word pair. Given a sequence of word representations $\mathbf{Z} = (\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_m) \in$

$\mathbb{R}^{m\times d}$, SAN compares all word pairs through a scaled dot-product function:

$$\begin{aligned}\mathbf{Q},\mathbf{K},\mathbf{V} &= f\left(\mathbf{Z}\right),\\ \text{Self-Attention}\left(\mathbf{Q},\mathbf{K},\mathbf{V}\right) &= \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right)\mathbf{V},\end{aligned} \tag{1}$$

where $f\left(\cdot\right)$ is a linear mapping function, d is the model dimension, and $\mathbf{Q},\mathbf{K},\mathbf{V}\in\mathbb{R}^{m\times d}$ are the corresponding queries, keys and values. The output of $softmax(\cdot)$ denotes the attention weights (a matrix of shape $m\times m$), which measures the degree of dependency between each word pair.[3] Often, we adopt the multi-head variant for SAN to enhance its expressivity (Vaswani et al., 2017).

When applying SAN to the decoder, we include a masking matrix into the above formulation, which disables the attention to future target words as required by the autoregressive schema. During training, Transformer with SAN is fully parallelizable; during inference, nevertheless, the autoregressiveness hinders the full parallelization since predicting the current target word must wait for the generation of all previous words. Our timing analysis shows that it is the decoder that dominates the running time (96.95%), of which SAN alone consumes about 41.42% – a significant decoding bottleneck. We thus explore how to make the decoder SAN computationally efficient.

3.2 Average Attention Network

As shown in Figure 1, the decoding of Transformer is inefficient partially due to its linear decoding dependency of the decoder SAN. This inspires researchers to develop efficient alternatives to SAN with simplified decoding dependency (Zhang et al., 2018a; Katharopoulos et al., 2020; Peng et al., 2021). In this paper, we focus on average attention network (`AAN`). Different from SAN, `AAN` functions at the j-th decoding step as follows:[4]

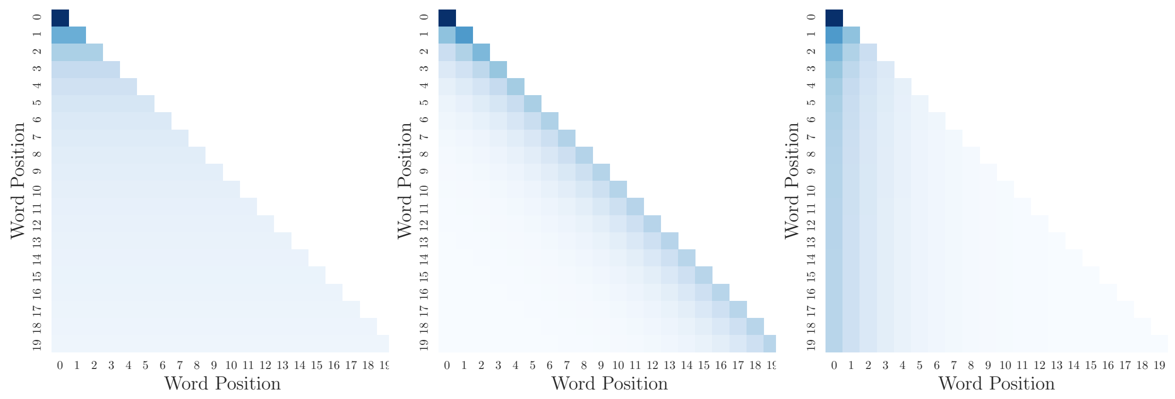
$$\mathbf{g}_j = \text{FFN}\left(\frac{1}{j}\sum_{k=1}^{j}\mathbf{z}_k\right), \tag{2}$$

$$\mathbf{i}_j,\mathbf{f}_j = \sigma\left(\mathbf{W}\left[\mathbf{z}_j;\mathbf{g}_j\right]\right),\quad \mathbf{o}_j = \mathbf{i}_j\odot\mathbf{z}_j + \mathbf{f}_j\odot\mathbf{g}_j, \tag{3}$$

where $\text{FFN}\left(\cdot\right)$ denotes the position-wise feed-forward network (Vaswani et al., 2017), $\mathbf{z}_k$, $\mathbf{g}_j$, and $\mathbf{o}_j$ have a dimension of d, $[\cdot;\cdot]$ denotes vector concatenation, $\odot$ indicates element-wise multiplication, and $\sigma(\cdot)$ is the sigmoid activation function.

The first part of `AAN`, as shown in Equation (2), employs a cumulative-average operation to generate a context-sensitive representation $\mathbf{g}_j$ for each target word. It replaces the dynamically calculated attention weights in SAN with the simple and fixed *average weights* $\{\frac{1}{j}\}$. The second part of `AAN`, as shown in Equation (3), fuses the previous context $\mathbf{g}_j$ and the word representation $\mathbf{z}_j$ to enrich the expressiveness of the model. Via a gating mechanism, `AAN` controls how much past information can be preserved from the context $\mathbf{g}_j$ and how much new information can be captured from the current word $\mathbf{z}_j$. In this way, `AAN` is allowed to detect dependencies inside the target sequence.

3. While attention weights could reflect the dependency, we also acknowledge that their interpretability is still debated in the literature (Kobayashi et al., 2020).
4. Note that `AAN` aims to replace SAN in the decoder and is *uni-directional*.



(a) Average Pattern (b) Neighboring Pattern (c) Distant Pattern

Figure 4: Visualization of the induced weights for different attention patterns. Deeper color indicates larger attention weights. Best viewed in color.

By decomposing Equation (2), `AAN` could compress all previous target information into one recurrent vector at inference, reducing the decoding dependency to $\mathcal{O}(1)$ compared to SAN. However, enforcing equal weights inevitably suffers from misleading and uninformative contexts, leading to suboptimal model representations. Recent studies show that local context is very useful for SAN in improving translation quality (Yang et al., 2018a), and our analysis on the learned attention weights of the decoder SAN also reveals attention patterns beyond the average as in Figure 2. It is thus preferable to improve the flexibility of `AAN` and extend it to support more diverse attention patterns. We explain how `AAN+` achieves this in the next section. To ease the description, we use $\mathbf{o}_j = \text{Gate}(\mathbf{z}_j, \mathbf{g}_j)$ for short to denote Equation (3).

4. Generalized Average Attention Network

At the core of `AAN` that enables the constant decoding dependency is its cumulative-style weight-sum structure. We discover that this structure can be further abstracted, according to which we propose generalized average attention network, `AAN+`. In particular, given a partial sequence $(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_j)$, `AAN+` decomposes the weight-sum formulation as follows:

$$\mathbf{g}_j = \sum_{k=1}^{j} w_k \mathbf{z}_k = \sum_{k=1}^{j} \frac{a_k}{\sum_{i=1}^{j} a_i} \mathbf{z}_k, \tag{4}$$

$$\mathbf{o}_j = \text{Gate}(\mathbf{z}_j, \mathbf{g}_j), \tag{5}$$

where each previous word representation $\mathbf{z}_k$ is associated with a weight w_k to reflect the dependency strength of the j-th word on the k-th word, and a_k is an unnormalized and positive score. Formally, a_k is regularized by the cumulative score up to j, such that,

$$w_k = \frac{a_k}{\sum_{i=1}^{j} a_i}, \text{ and } \sum_{k=1}^{j} w_k = 1.$$

We abandon the FFN(·) part in Equation (2) because FFN brings in a large number of model parameters while contributing little to the final translation quality, as demonstrated in (Zhang et al., 2018a).

This decomposition opens the possibility to support more diverse attention patterns beyond the average one. To retain the decoding dependency of `AAN`, `AAN+` requires each score a_k to be estimated independently from the information at other positions, i.e.

$$a_k = \delta(k, \mathbf{z}_k). \tag{6}$$

Equation (6) describes the high-level structure of the attention supported by `AAN+`. On one hand, $\delta(\cdot)$ avoids the dependency of $\mathbf{z}_k$ on any previous word representation such that the decoding dependency can still be reduced to $\mathcal{O}(1)$ (see Section 4.2). On the other hand, $\delta(\cdot)$ provides the flexibility in manipulating the attention distribution, allowing us to consider different information and heuristic rules.

Inspired by the analysis in Figure 2, we mainly study the following four patterns, taking into account either content or position (shown in Figure 4) information:

***Average Pattern* (Avg)** Treating each word equally (i.e., following a uniform prior) has benefited various NLP tasks, such as word embedding (Mikolov et al., 2013), sentence classification (Zhang et al., 2015) and machine translation (Sutskever et al., 2014; Zhang et al., 2018a). This pattern adopts a position- and content-agnostic scoring function that shares the same spirit as `AAN`:

$$\delta_{\text{AVG}}(k, \mathbf{z}_k) = 1. \tag{7}$$

This definition also reveals that `AAN` can be regarded a special case of `AAN+`.[5]

***Neighboring Pattern* (Ner)** Figure 2 shows that some attention heads in the decoder SAN tend to focus on neighboring words. Also, recent studies show the importance of local context or nearby words in machine translation (Yang et al., 2018a; Luong et al., 2015). We thus develop NER, which is position sensitive and puts more distribution mass on neighboring words:

$$\delta_{\text{NER}}(k, \mathbf{z}_k) = \exp(\alpha k), \tag{8}$$

where $\alpha \in (0, 1)$ is a hyperparameter used to control the sharpness of the distribution, which also improves the training stability by avoiding value explosion.

***Distant Pattern* (Far)** We observe that a large proportion of attention heads in the decoder SAN prefer to fuse information from the beginning or distant words, particularly at upper layers as in Figure 2. We argue that distant words might have larger impact on the current word representation, especially when a long-distance dependency structure occurs. Based on this insight, we exploit FAR which operates in the opposite of NER as follows:

$$\delta_{\text{FAR}}(k, \mathbf{z}_k) = \exp(-\beta k), \tag{9}$$

where the negative sign inside the exponential function endows a larger position with a smaller score, and $\beta \in (0, 1)$ behaves similarly to α in NER.

5. Note that *AAN+ w/* AVG is not the same as `AAN` due to the drop of the FFN(·) layer. But they are equivalent in the sense that they share the same equal-weight assumption.

$$\underset{\substack{\mathrm{M}\\ \text{Mask Matrix}}}{\begin{pmatrix}1&0&0&0\\1&1&0&0\\1&1&1&0\\1&1&1&1\end{pmatrix}} \times \underset{\mathrm{E}}{\begin{pmatrix}\mathrm{e}_1\\\mathrm{e}_2\\\mathrm{e}_3\\\mathrm{e}_4\end{pmatrix}} = \underset{\mathrm{ME}}{\begin{pmatrix}\mathrm{e}_1\\\mathrm{e}_1+\mathrm{e}_2\\\mathrm{e}_1+\mathrm{e}_2+\mathrm{e}_3\\\mathrm{e}_1+\mathrm{e}_2+\mathrm{e}_3+\mathrm{e}_4\end{pmatrix}}$$

Figure 5: Illustration of the parallel implementation for the cumulative-sum operation by applying a mask matrix $\mathbf{M}$ to four word representations $\mathbf{E} = [\mathbf{e}_1; \mathbf{e}_2; \mathbf{e}_3; \mathbf{e}_4]$.

Weighted Pattern **(Wet)** One appealing property SAN possesses is that the attention weights are learned automatically via neural networks. This gives the model the capability of adjusting weights to make them suitable for different inputs. In fact, many attention heads in the decoder SAN show no clear position-related patterns in Figure 2, which are likely content-dependent. Following this spirit, we develop Wet as follows:

$$\delta_{\text{Wet}}(k, \mathbf{z}_k) = \exp(\gamma \mathbf{U}\mathbf{z}_k), \tag{10}$$

where $\mathbf{U} \in \mathbb{R}^{d\times d}$ is the weight parameter that projects word representations into a fine-grained scoring space, and γ performs similarly to α and β. We expect that this linear mapping can strengthen `AAN+`'s capacity in modeling token-wise dependency.

Note that the above-mentioned hyperparameters α, β and γ are optimized by grid search based on the performance on development set.[6] Figure 4 visualizes different patterns except for Wet because it relies on inputs.[7] Different patterns reflect different modeling considerations, which are carefully compared via experiments. In the following sections, we use `AAN+`$(\cdot)$ to refer to the whole procedure formulated in Equation (4-5).

4.1 Parallelization in Training

One computational bottleneck of `AAN+` is that its cumulative-sum operation in Equation (4) can only be performed sequentially. In other words, this operation is not parallelizable. We solve this problem by using a masking trick as shown in Figure 5, where the word representations are directly converted into their corresponding cumulative-sum outputs through a masking matrix. Accordingly, the formulation in Equation (4) can be rewritten as follows:

$$\mathbf{G} = \frac{\mathbf{M}(\mathbf{A} \odot \mathbf{Z})}{\mathbf{MA}}, \tag{11}$$

where $\mathbf{Z} \in \mathbb{R}^{m\times d}$ is the word representation matrix. $\mathbf{A}$ denotes the score matrix, which can be a coarse-grained score matrix of shape $\mathbb{R}^{m\times 1}$ or a fine-grained score matrix of shape $\mathbb{R}^{m\times d}$, depending on the choice of the scoring function. $\mathbf{M} \in \mathbb{R}^{m\times m}$ is a lower triangular

6. We also attempted to automatically learn α and β, i.e. let the model induce different patterns on its own. Unfortunately, optimization is non-trivial and is hindered by the *NaN* issue.
7. When drawing the figure, we set α and β to 0.35 to improve the clarity and contrast.

Model	**SAN**	**Sequential** `AAN+`	**Masked** `AAN+`
Complexity	$\mathcal{O}\left(m^2 \cdot d + m \cdot d^2\right)$	$\mathcal{O}\left(m \cdot d^2\right)$	$\mathcal{O}\left(m^2 \cdot d + m \cdot d^2\right)$
Sequential Operations	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$
Maximum Path Length	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
Decoding Dependency	$\mathcal{O}(m)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$

Table 1: Maximum path length, model complexity, decoding dependency, and minimum number of sequential operations for different models. m is the sentence length and d is the model size. Model complexity is an indicator of the training-time complexity; decoding dependency, by contrast, corresponds to the decoding-time complexity.

masking matrix.[8] In this way, all components inside `AAN+` can yield full parallelization, assuring its computational efficiency particularly on GPU. We refer to this `AAN+` as *masked* ***AAN+***, and the original `AAN+` as *sequential* ***AAN+***.

4.2 Decoding Acceleration

When applying to the decoder of Transformer, `AAN+` can be accelerated during decoding via dynamic programming because of the independence constraint of $\delta(\cdot)$. We decompose Equation (4) into the following three steps:

$$\tilde{\mathbf{g}}_j = \tilde{\mathbf{g}}_{j-1} + a_j \mathbf{z}_j, \tag{12}$$

$$\tilde{a}_j = \tilde{a}_{j-1} + a_j, \tag{13}$$

$$\mathbf{g}_j = \frac{\tilde{\mathbf{g}}_j}{\tilde{a}_j}, \tag{14}$$

where $\tilde{\mathbf{g}}_0 = \mathbf{0}, \tilde{a}_0 = 0$. In doing so, our model computes the j-th word representation based on only one recurrent state $\tilde{\mathbf{g}}_{j-1}$, instead of relying on all previous states as SAN does. Therefore, our model can be substantially accelerated at the decoding phase.

4.3 Model Analysis

In this subsection, we provide a thorough analysis for `AAN+` in comparison to SAN. Following Vaswani et al. (2017), we compare both models in terms of computational complexity, the minimum number of sequential operations required and the maximum path length between any two input and output positions that forward and backward signals have to traverse in the network.

Table 1 summarizes the comparison results. `AAN+` has a maximum path length of $\mathcal{O}(1)$, because each output representation is directly connected with each previous input representation. For *sequential* `AAN+`, its computation avoids the position-wise masked projection which reduces its computational complexity to $\mathcal{O}\left(m \cdot d^2\right)$, but enlarges its minimum number of sequential operations to $\mathcal{O}(m)$. By contrast, both SAN and *masked* `AAN+` have a

8. This masking matrix also applies to the vanilla decoder SAN to disable the access to future target words.

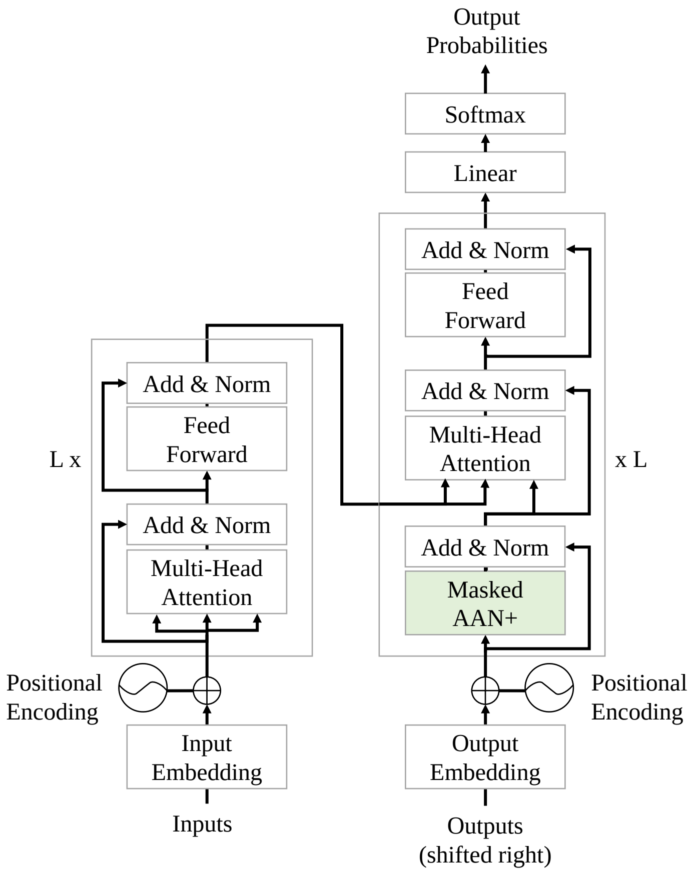


Figure 6: The new Transformer architecture with the proposed AAN+

computational complexity of $\mathcal{O}\left(m^2 \cdot d + m \cdot d^2\right)$, and require only $\mathcal{O}\left(1\right)$ sequential operation. Theoretically, *masked* AAN+ performs very similarly to SAN according to Table 1, but with the superiority of short decoding dependency of $\mathcal{O}\left(1\right)$. We therefore use *masked* AAN+ for training throughout all our experiments.

5. Neural Transformer with AAN+

We apply our *masked* AAN+ to Transformer – the state-of-the-art encoder-decoder model for sequence-to-sequence tasks (Vaswani et al., 2017). We retain the overall structure of Transformer and only replace its decoder SAN with AAN+ as shown in Figure 6.

Given a source sequence $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$, Transformer leverages its encoder to learn source-side semantics and dependencies, and output contextualized word representations to offer clues for its decoder. The encoder is composed of a stack of $L = 6$ identical layers, each of which has two sub-layers:

$$\begin{aligned}\tilde{\mathbf{H}}^l &= \text{LayerNorm}\left(\mathbf{H}^{l-1} + \text{MHAtt}\left(\mathbf{H}^{l-1}, \mathbf{H}^{l-1}, \mathbf{H}^{l-1}\right)\right), \\ \mathbf{H}^l &= \text{LayerNorm}\left(\tilde{\mathbf{H}}^l + \text{FFN}\left(\tilde{\mathbf{H}}^l\right)\right),\end{aligned} \tag{15}$$

where the superscript l indicates the layer depth, and $\mathrm{MHAtt}(\cdot,\cdot,\cdot)$ denotes the multi-head attention mechanism proposed by Vaswani et al. (2017).

Based on the contextualized representations $\mathbf{H}^L \in \mathbb{R}^{n\times d}$, Transformer uses its decoder to generate the corresponding target sequence $\mathbf{Y} = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_m)$. The decoder also consists of a stack of $L = 6$ identical layers, each of which involves three sub-layers. The first sub-layer aims at capturing target-side dependency on previously predicted words. Rather than using SAN ($\overline{\mathrm{MHAtt}}$ denotes the masked variant that disables future attention)

$$\tilde{\mathbf{S}}^l = \mathrm{LayerNorm}\left(\mathbf{S}^{l-1} + \overline{\mathrm{MHAtt}}\left(\mathbf{S}^{l-1}, \mathbf{S}^{l-1}, \mathbf{S}^{l-1}\right)\right), \tag{16}$$

we adopt the proposed `AAN+` to improve inference efficiency:

$$\tilde{\mathbf{S}}^l = \mathrm{LayerNorm}\left(\mathbf{S}^{l-1} + \texttt{AAN+}\left(\mathbf{S}^{l-1}\right)\right). \tag{17}$$

With the learned dependency, the decoder further stacks an encoder-decoder attention sub-layer to detect prediction-relevant source information, followed by a feed-forward sub-layer:

$$\begin{aligned}\mathbf{S}_c^l &= \mathrm{LayerNorm}\left(\tilde{\mathbf{S}}^l + \mathrm{MHAtt}\left(\tilde{\mathbf{S}}^l, \mathbf{H}^L, \mathbf{H}^L\right)\right), \\ \mathbf{S}^l &= \mathrm{LayerNorm}\left(\mathbf{S}_c^l + \mathrm{FFN}\left(\mathbf{S}_c^l\right)\right).\end{aligned} \tag{18}$$

We use the subscript c to denote the source-informed target representation. The generation is performed upon the top decoder layer, where a linear transformation and the softmax activation are applied to get the probability of the next token based on $\mathbf{S}^L \in \mathbb{R}^{m\times d}$.

To make the learned representations position-sensitive, Transformer augments its input layer $\mathbf{H}^0 = \mathbf{X}, \mathbf{S}^0 = \mathbf{Y}$ with frequency-based positional encodings. The whole model is a large, single neural network, and can be optimized end-to-end with the maximum likelihood objective. We refer readers to (Vaswani et al., 2017) for more details.

5.1 Decoding Acceleration Analysis

`AAN+` has a shorter decoding dependency ($\mathcal{O}(1)$) than SAN ($\mathcal{O}(m)$). However, this doesn't imply that `AAN+` translates m times faster. The reason lies at that `AAN+` or SAN is just one component of the decoder as shown in Figure 6. In this subsection, we show theoretical analysis explaining how reducing the decoding dependency on previously generated words accelerates the generation.

We mainly analyze the one-step decoding in Transformer, and consider the use of *cache* where the keys and the values in attention networks are pre-calculated or maintained during decoding. `AAN+` and SAN share the same encoder-decoder attention sub-layer and feed-forward sub-layer, with a decoding complexity of $\mathcal{O}(n\cdot d + d^2)$ and $\mathcal{O}(d^2)$, respectively. In total, the decoder with SAN has a decoding complexity of $\mathcal{O}(m\cdot d + n\cdot d + d^2)$, while `AAN+` reduces it to $\mathcal{O}(n\cdot d + d^2)$. This result suggests that a longer target sequence compared to its source counterpart leads to a larger decoding speedup with a ratio of around $\frac{m}{n+d} + 1$. That is, the acceleration effect highly depends on the target-source length ratio and the model size.

As the model size (such as 512 or 1024 in Transformer) is often larger than the sequence length, the acceleration of `AAN+` is marginal in theory. Nevertheless, in practice, `AAN+`

avoids not only maintaining all previous target hidden states (the so-called *cache*) but also performing the softmax and concatenation operations. These operations often hinder the modern hardware from fully parallelizable computation. For example, the computations after the SAN softmax operation will be blocked until it is finished, causing delay. In this respect, `AAN+` improves decoding parallelization and saves running memory, which works even for large models (Junczys-Dowmunt et al., 2018).

6. Experiments

To examine the effectiveness of our model, we conduct experiments on machine translation with several language pairs. We also work on the table-to-text generation task and the document summarization task, aiming at examining the generality of `AAN+` and its ability in modeling long-range dependency. Unless otherwise specified, all experiments are performed with GeForce GTX 1080 Ti and all models are implemented in Tensorflow.

6.1 Data

We choose three translation tasks, including WMT14 English-German translation (En-De, Bojar et al., 2014), WMT14 English-French translation (En-Fr) and NIST Chinese-English translation (Zh-En, Zhang et al., 2020). These translation tasks cover high-resource, medium-resource and *relatively* low-resource settings, as well as language pairs with diverse dependency orders. We select WMT14 En-De as our benchmark for analysis, and use the same WMT14 dataset as in (Vaswani et al., 2017). The training data have 4.5M/36M/1.25M sentence pairs, involving ∼116M/1045M/28M source words and 110M/1189M/35M target words for WMT14 En-De/WMT14 En-Fr/NIST Zh-En, respectively.

We use newstest2013, newstest2012+2013 and NIST2005 as the development set for WMT14 En-De, WMT14 En-Fr and NIST Zh-En, respectively. We report test results on newstest2014 for WMT14 En-De and WMT14 En-Fr, and NIST2002, NIST2003, NIST2004, NIST2006, NIST2008 for NIST Zh-En. We apply the byte pair encoding (BPE) algorithm (Sennrich et al., 2016) with 32K merging operations to handle out-of-vocabulary tokens for all these tasks. We employ tokenized case-sensitive BLEU (Papineni et al., 2002) calculated using *multi-bleu.perl*[9] as well as METEOR (Denkowski & Lavie, 2014)[10] to evaluate translation quality.

We adopt the *WIKIBIO* dataset (Lebret et al., 2016) for table-to-text generation. This dataset contains 728,321 articles from English Wikipedia (Sep, 2015), with each article containing the first paragraph and an infobox. The infobox follows a key-value structure, such as "*Born: 12 November 1914*" (from Figure 1 in (Lebret et al., 2016)). We convert each infobox into a sequence with the format of key_1 $value_1$ key_2 $value_2$ … key_n $value_n$, regardless of its structure, and generate the first paragraph from this sequential infobox. We obtain 582,659 articles for training, 72,831 articles for validation, and another 72,831 articles for testing. We pre-process these articles using BPE with 20K merging operations, and measure the generation quality with BLEU-4 and ROUGE-4 (F1 score).[11]

9. `https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl`
10. We used the METEOR library at `https://www.cs.cmu.edu/~alavie/METEOR/download/meteor-1.5.tar.gz`. Models are evaluated via "java -Xmx2G -jar meteor-1.5.jar *test reference* -l *lang* -lower".
11. Following previous work, we use *NIST mteval-v13a.pl* for BLEU and *MSR rouge-1.5.5* for ROUGE.

We use the *CNN/Daily Mail* dataset (Hermann et al., 2015) for document summarization. This dataset includes summaries that are bullet points for the articles from their corresponding websites. Following Gehrmann et al. (2018), we use the non-anonymized version of this corpus and truncate each source document to 400 tokens and each target summary to 100 tokens in the training and validation sets. We have 287,227 document-summary pairs in the training set, 13,368 pairs and 11,490 pairs in the validation set and test set, respectively. Sequences are processed via BPE with 32K merging operations. We employ neither copy mechanism nor pointer network, because Transformer with BPE alone works well, delivering comparable or even better performance than previously reported results. We report ROUGE (ROUGE-1, ROUGE-2, Rouge-L) for summarization evaluation.

6.2 Model Setup

By default, we adopt the base setting (Vaswani et al., 2017): the model size d is 512, the middle layer size of FFN($\cdot$) is 2048, and the head number is 8. We use Adam (Kingma & Ba, 2015) ($\beta_1 = 0.9$, $\beta_2 = 0.98$, and $\epsilon = 10^{-8}$) to train model parameters, with a batch size of roughly 25000 target tokens.[12] We schedule the learning rate using the inverse square root of running steps, with a warmup step of 4000. To avoid over-fitting, we employ label smoothing with $\epsilon_{ls} = 0.1$, attention dropout and residual dropout with a rate of $p = 0.1$. We use the beam search algorithm for decoding and set the beam size to 4 and the length penalty to 0.6. We set α, β and γ in `AAN+` to 0.1 according to our preliminary experiments, and study their impact on translation performance in Section 6.3.1. We train the models for around 300K steps (more or less for different training datasets of diverse scales), and average the last five checkpoints saved with an interval of 5000 training steps for evaluation.[13]

Apart from the vanilla Transformer, we further compare our method with two other baselines:

SSRU This is a Simplified simple recurrent unit (SSRU) that reduces the amount of matrix computations and contains just a simple gating function in its recurrent transition, which gains inference efficiency significantly compared to SAN (Kim et al., 2019).

Linear Attention This model reduces the complexity of SAN from $\mathcal{O}(m^2)$ to $\mathcal{O}(m)$ by leveraging the associativity property of matrix products. Katharopoulos et al. (2020) reported up to 4000x speedup on autoregressive prediction of very long sequences.

Note we ***re-implement*** all these baselines in Tensorflow based on our in-house codebase, and set their model dimension to 512 by default in our experiments.

6.3 WMT14 English-German Translation

Table 2 summarizes the results. Our Transformer baseline achieves a BLEU score of 27.59 on Test14, slightly better than the reported BLEU in (Vaswani et al., 2017) (+0.29). Replacing SAN in the decoder with `AAN+` results in very marginal decrease in the overall translation quality, where the maximum drop is around 0.24 BLEU and 0.20 METEOR delivered by

12. For each batch, we add sentence pairs until the next sentence pair would exceed 25000 target tokens.
13. We set the number of training steps for each task based on our experience, and stabilize the model evaluation via checkpoint averaging.

Model	**#Params**	**△Train**	**△Decode**	**BLEU**	**METEOR**
Transformer	72.31M	1.00×	1.00×	**27.59**	47.93
SSRU	69.16M	0.85×	1.35×	27.55	47.56
Linear Attention	72.31M	0.51×	0.22×	27.62	47.65
`AAN+` w/ Avg	72.30M	**1.03**×	**1.32**×	27.50	47.80
`AAN+` w/ Far	72.30M	1.02×	1.28×	27.35	47.73
`AAN+` w/ Ner	72.30M	1.02×	1.30×	27.56	47.81
`AAN+` w/ Wet	73.88M	0.97×	1.16×	27.49	**48.01**

Table 2: Case-sensitive tokenized BLEU and METEOR on WMT14 En-De. "*#Params*": the number of trainable parameters. "△*Train*": training speedup evaluated on 0.2K steps with a batch size of 25K target tokens. "△*Decode*": decoding speedup on Test14 with a batch size of 32. We perform 5 different runs and report the average results. "*M*": million. Higher BLEU↑ and METEOR↑ indicate better translation quality. The best results are highlighted in **bold**.

Far. Besides, Wet even matches SAN in METEOR (+0.08). This suggests the structure redundancy of the decoder SAN in Transformer and the feasibility of simplifying it.

Different variants of `AAN+` show different properties: the position-based patterns (Ner, Avg) perform slightly better than the content-based one (Wet) in BLEU; while this relation is reversed when considering METEOR, a metric more consistent with human judgement (Barrault et al., 2019). In terms of BLEU, Ner performs the best (27.56) among them, followed by Avg (27.50) and Wet (27.49). Far performs the worst (-0.24 BLEU and -0.20 METEOR compared to the baseline). The preference for neighboring tokens over distant in `AAN+` (+0.21 BLEU and +0.08 METEOR, Ner vs. Far) shows the usefulness of local information to sequence generation, resonating with previous findings (Yang et al., 2018a; You et al., 2020; Raganato et al., 2020).

Although the adaptive weighting schema endows Wet with higher freedom in describing token-wise dependencies, Wet fails to improve the translation quality against the baseline under different metrics (+0.08 METEOR but -0.10 BLEU). In addition, Wet even performs slightly worse than Avg by 0.01 BLEU. The JS-divergence curve in Figure 7 suggests that Wet fails to mimic the behavior of SAN, particularly at the top and bottom decoder layers. Inspired by this observation, we performed a further experiment that replaces Wet with the vanilla SAN in the first decoder layer. This narrows the BLEU reduction against the baseline to 0.03 and retains the gain in METEOR (+0.05). Thus, we ascribe the inferior performance of Wet to its insufficient modeling capability resulted from the cumulative operation which cannot fully capture word-pair correlations as in the dot-product attention.

`AAN+` marginally reduces the amount of parameters (-0.01M) and improves the training speed (∼1.02× faster), except Wet which adds a linear mapping to generate input-dependent scores. Overall, no significant computational overhead is brought in by `AAN+` at the training phase. Regarding translation, `AAN+` accelerates the decoding with a speedup of

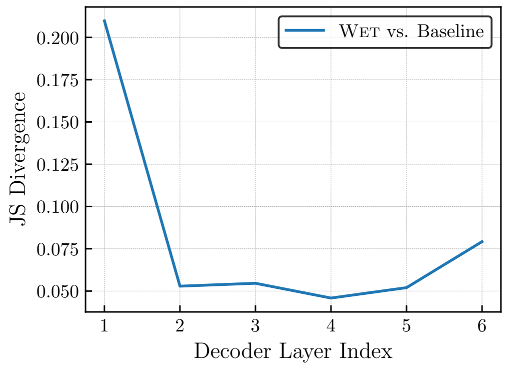


Figure 7: Jensen-Shannon (JS) divergence (y-axis) of the decoder-side self-attention weights between Wet and Baseline across layers (x-axis). For each layer, we average the weights over either dimensions (Wet) or heads (Baseline) before computing this divergence. We report the average score over all target tokens on WMT14 En-De (newstest2014).

$1.16\times \sim 1.32\times$.[14] Note that Far and Ner translate slower than Avg, though they all are position-based and follow a similar structure. We conjecture that the denominator $\sum_i^j a_i$ in Equation (4), which has to be explicitly maintained in Far and Ner during translation, increases the memory usage and hinders model parallelization. This leads to slower decoding, and also resonates with our analysis in Section 5.1 that simplified recurrent computation is beneficial for reducing running time.

Table 2 also shows the results for SSRU and Linear Attention. SSRU contains fewer model parameters with reduced computations, and Linear Attention simplifies the computational complexity of SAN to $\mathcal{O}(m)$. Despite these advantages, they run significantly slower at training. The linear computational complexity of Linear Attention comes at the cost of sacrificing its space complexity $\mathcal{O}(md^2)$, which demands substantially higher running memory; SSRU is basically a recurrent model that requires dedicated CUDA kernels to reach decent training efficiency. In contrast, `AAN+` is simple to implement and fast to train; some operations like *cumsum* are often well supported by popular computational frameworks.

Although Linear Attention matches the baseline in BLEU, it delivers significantly worse inference speed ($0.22\times$) different from the findings in (Katharopoulos et al., 2020). The reason behind, we argue, is that Linear Attention relies on $\mathcal{O}(d)$ recurrent vectors during translation that only improves decoding speed when the target sequence is extremely long. By contrast, SSRU yields an encouraging decoding speedup of $1.35\times$ thanks to its reduced computation (Kim et al., 2019). However, the translation quality of SSRU is inferior to `AAN+` and the baseline as measured by METEOR.

6.3.1 Impact of `AAN+` Hyperparameters

`AAN+` uses the hyperparameters α, β and γ to control the shape of its attentional distribution for generation. Figure 8 shows their impact on translation. We observe that these

14. We reported roughly $4\times$ speedup in our conference paper (Zhang et al., 2018a), which is because we didn't apply the *cache*-based decoding to the Transformer baseline. Nowadays, using cache for decoding has already been a standard practice. The numbers reported in this paper are more fair and accurate.

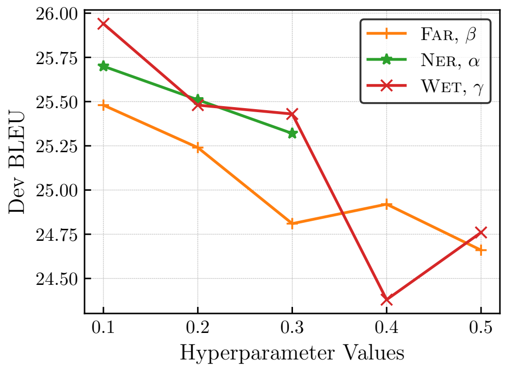


Figure 8: Translation performance as a function of α (Ner), β (Far) and γ (Wet) on the WMT14 En-De dev set (newstest2013). We vary these hyperparameters from 0.1 to 0.5 with a step size of 0.1. Each model is trained for 100K running steps. Note we didn't report results for Ner with $\alpha = 0.4/0.5$ because of the NaN issue at training.

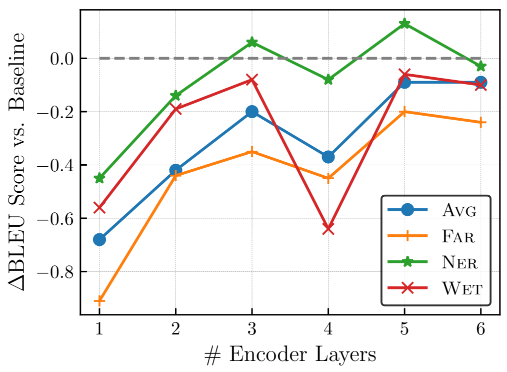


Figure 9: Case-sensitive tokenized BLEU (y-axis, vs. the baseline) of different models on WMT14 En-De (newstest2014) with different encoder depths (x-axis). Dashed gray line illustrates the SAN baseline. Note that all `AAN+` models suffer from a larger performance drop when reducing the encoder depth compared to the baseline, among which Ner shows the best robustness.

hyperparameters greatly affect the translation quality, where larger values often lead to worse performance. Importantly, setting these hyperparameters properly also stabilizes model training, avoiding the NaN issue for Ner. Based on the above results, we set these hyperparameters to 0.1 for all other experiments in this paper.

6.3.2 Why Does `AAN+` Work?

Table 2 shows that `AAN+` yields roughly comparable performance against the multi-head dot-product attention in spite of its simplicity. One assumption is that the success of `AAN+` is supported by Transformer through shifting more modeling burden to the encoder. To examine this, we vary the encoder depth for both `AAN+` and the SAN baseline and report

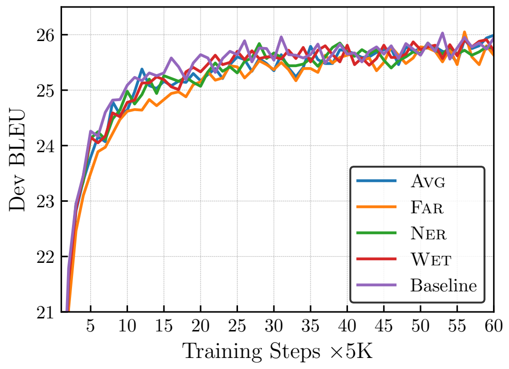
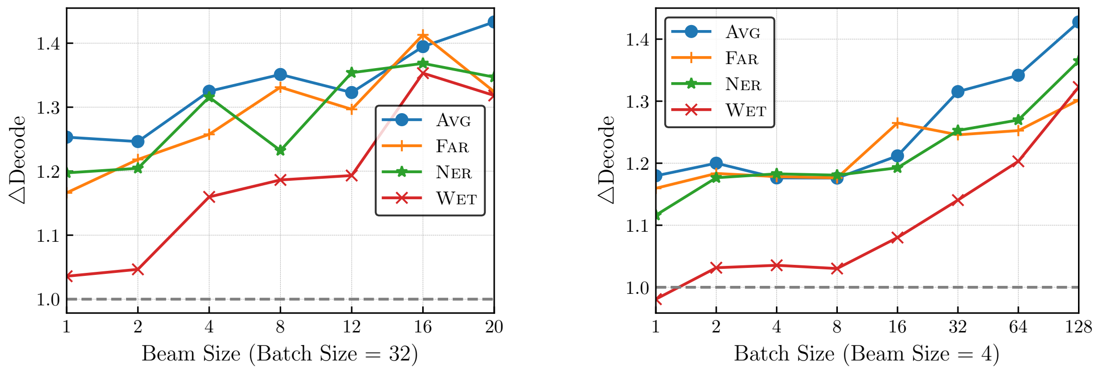


Figure 10: Case-sensitive tokenized BLEU (y-axis) of different models on the WMT14 En-De dev set (newstest2013) over training steps (x-axis). `AAN+` converges similarly to the Transformer baseline.



(a) Beam Effect (b) Batch Effect

Figure 11: Impact of beam size (left) and batch size (right) on decoding acceleration (△Decode, y-axis). Statistics are collected from one run on the WMT14 En-De test set. `AAN+` achieves higher decoding speedups with larger beam sizes and/or larger batch sizes, and position-based patterns perform better than Wet.

their relative BLEU scores in Figure 9. The results partially confirm our assumption, where decreasing the encoder depth to 1 causes ∼0.65 extra BLEU reduction (on average). Nevertheless, this performance drop is not that aggressive, further suggesting that the intra-sentence dependency in the decoder requires less modeling capacity, allowing for simplified alternatives (You et al., 2020; Kasai et al., 2021). Also note that Ner dominates the performance of `AAN+`. Modeling local context is crucial to the decoder for generation (Luong et al., 2015; Yang et al., 2018a).

6.3.3 Analysis on Model Convergency

One potential drawback of model simplification, like `AAN+`, is delaying the model convergence. Figure 10 compares the convergence of different models with respect to BLEU on

Device	Baseline	Avg	Far	Ner	Wet
GPU	61.60 (1.00×)	46.62 (**1.32**×)	48.18 (1.28×)	47.34 (1.30×)	53.12 (1.16×)
CPU	663.87 (1.00×)	474.97 (**1.40**×)	520.10 (1.28×)	508.33 (1.31×)	535.39 (1.24×)

Table 3: Decoding time in seconds (speedup) on Test14 with different devices (CPU vs. GPU). Our CPU is *Intel(R) Xeon(R) CPU E5-2680 0 @ 2.70GHz* supporting 32 processors.

the development set. `AAN+` and the baseline show highly similar convergence behaviour and achieve almost the same BLEU score, although the performance of `AAN+` increases a little bit slower than Transformer at the beginning (Far in particular).

6.3.4 Analysis on Decoding Speed

We provide a theoretical analysis in Section 5.1 on decoding acceleration. In practice, however, many factors could affect the running speed. In this subsection, we further explore the effect of beam size, batch size and GPU/CPU on the decoding speed.

Beam size determines how many top-scored hypothesises coexist for one source sentence during translation. Decoding with a larger beam size under proper regularization can enhance translation quality but also bring in non-negligible computational overheads. Figure 11a shows that `AAN+` results in better acceleration when using larger beam sizes ($1.16\times \rightarrow 1.36\times$ with beam size $1 \rightarrow 20$ on average). The increase of recurrent computations under larger beams enhances the strength of `AAN+`.

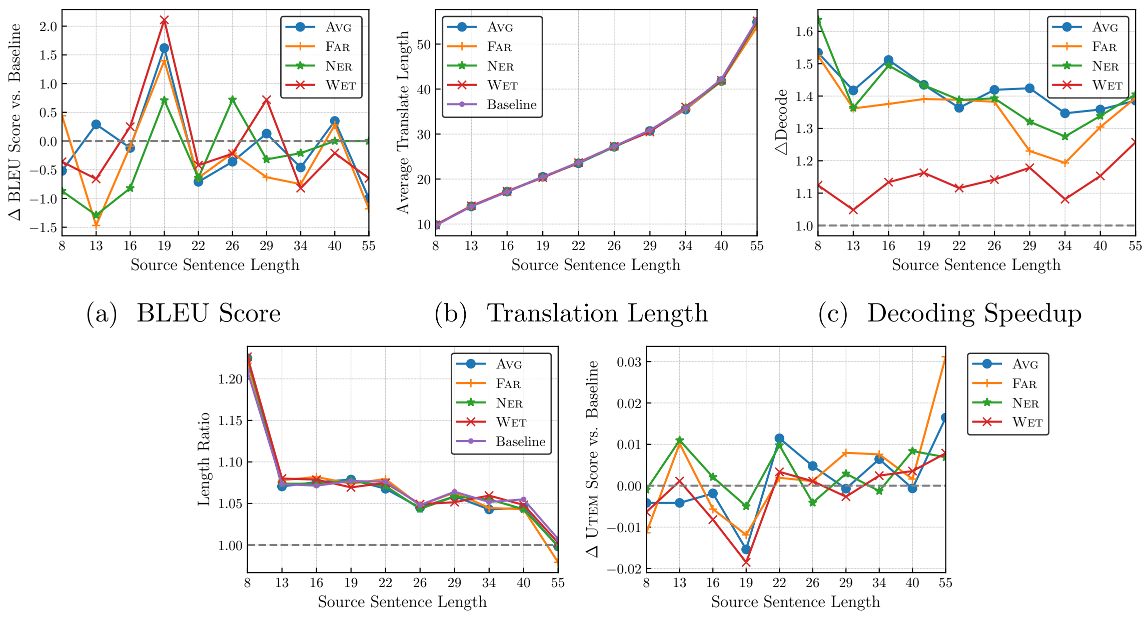
Batch size denotes the number of source sentences translated simultaneously. In our experiments, we group sentences of similar length into one batch for decoding. This maximizes the utilization of those modern hardware devices, like GPU, and saves the overall translation time. Results in Figure 11b indicate that `AAN+` favours larger batch sizes ($1.11\times \rightarrow 1.35\times$ with batch size $1 \rightarrow 128$ on average). Particularly, `AAN+` accelerates the translation with a batch size of 1, showing its potential of improving online translation services in industries.

In addition, Figure 11 also shows that position-based patterns outperform the content-based one (Wet) in decoding speed. Far and Ner run slightly slower than Avg.

Device also affects the decoding process. Although CPU performs worse than GPU on large-matrix multiplication, it supports some operations like discrete softmax and memory copying better. Table 3 shows that `AAN+` is capable of accelerating decoding regardless of the device types, where Avg yields the best speedup, followed by Ner. We also observe that `AAN+` is more CPU-friendly: its speedup on CPU is often larger than that on GPU across different patterns.

6.3.5 Effects of Sentence Length

NMT often struggles with long sentence translation as handling long-distance dependency and under-translation issues becomes more difficult for longer sentences. In this section, we investigate whether all patterns in `AAN+` are capable of dealing with long-range dependency. We experiment on newstest2014, where we first sort sentence pairs according to their source



(a) BLEU Score (b) Translation Length (c) Decoding Speedup

(d) Length Ratio (e) UTEM Score

Figure 12: Analysis of translations on WMT14 En-De (newstest2014) with respect to the length of source sentences (x-axis). (a) shows the tokenized BLEU, (b) shows the average length of translations, (c) shows the decoding speedup over the baseline, (d) shows the ratio between the average translation length and the average source length, and (e) shows the case-sensitive tokenized UTEM, all visa-vis source sentence length. The number in x-axis denotes the average source sentence length in its corresponding evaluation subset. Lower UTEM↓ and higher BLEU↑ indicate better translation.

sentence lengths, and then split the dataset into 10 disjoint subsets of roughly equal size (about 274 sentence pairs each).[15] We re-evaluate different measures on these subsets.

Figure 12 shows the results. Although the adaptive dot-product based SAN is often deemed more effective than the average or rule-based attentions (`AAN+`) in generating (long) sentences, our results suggest that simple attentions (`AAN+`) are able to produce translations of similar BLEU score (Figure 12a) and length (Figure 12b) compared to SAN. Besides, Transformer fails to outperform `AAN+` over all different source sentence lengths (Figure 12a), which partially verifies the ability of average patterns in capturing long-range dependency. Note that Ner matches the performance of Transformer on the longest sentence group, though this pattern schedules more distribution masses to local (neighbouring) context.

Generally, translating longer sentences consumes more decoding time: when it comes to Transformer, this issue becomes notably severe since all previous predicted words must be considered to estimate the self-attention weights. As an important factor, source sentence length deeply affects the acceleration of `AAN+` as discussed in Section 5.1. Here, we further

15. Note that this differs from what we did in our conference paper, where we collected sentence pairs based on certain length range. The problem is that the obtained subsets are highly imbalanced. Evaluation on subsets including only a few sentence pairs can be very inaccurate and misleading.

Settings	Model	#Params	△Decode	BLEU	METEOR
Deep Transformer	Transformer	116.45M	1.00×	28.27	**48.49**
	SSRU	110.14M	1.38×	27.87	48.06
	Linear Attention	116.45M	0.19×	28.22	48.41
	AAN+ w/ Avg	116.44M	**1.40**×	28.29	48.39
	AAN+ w/ Far	116.44M	1.39×	28.17	48.35
	AAN+ w/ Ner	116.44M	1.39×	**28.35**	48.29
	AAN+ w/ Wet	119.59M	1.22×	28.31	48.21
Big Transformer	Transformer	232.70M	1.00×	**29.09**	**48.95**
	SSRU	220.10M	1.30×	28.63	48.65
	Linear Attention	232.70M	0.21×	28.85	48.86
	AAN+ w/ Avg	232.69M	1.30×	28.79	48.58
	AAN+ w/ Far	232.69M	1.29×	28.76	48.55
	AAN+ w/ Ner	232.69M	1.29×	28.94	48.77
	AAN+ w/ Wet	238.99M	**1.38**×	28.87	48.64

Table 4: Case-sensitive tokenized BLEU and METEOR for different models on WMT14 En-De (newstest2014). *Deep Transformer*: 12-layer Transformer base model; *Big Transformer*: 6-layer Transformer with $d = 1024$, 16 heads and FFN size of 4096. Decoding time is averaged over five runs; the time for Big Transformer is measured on GeForce GTX 1080.

offer empirical results. Figure 12c suggests that AAN+ accelerates translation regardless of source sentence lengths with a special peak at the shortest sentence group.[16] Figure 12d tells the reason, where the shortest sentence group has the largest target-source length ratio, confirming our analysis in Section 5.1.

In addition, NMT models often suffer from the *under-translation* issue (Zhang et al., 2018a), where important source tokens or phrases are totally ignored during translation. This could become worse if the model can't properly handle long-distance context. We employ the under-translation evaluation metric (Yang et al., 2018b, Utem) to measure this issue, which counts how many n-grams are missed in the translation compared with its reference. Figure 12e shows that AAN+ yields comparable performance against Transformer. On long sentences, however, AAN+ obtains worse Utem scores. We argue that AAN+ is capable of capturing long-range dependency, but its capacity is still weaker than SAN. Especially, despite of its localness, Ner performs better than Avg on the longest sentence group.

6.3.6 Results for Deep and Big Models

We also extend our experiments to deep and big settings. For deep Transformer, we adopt the depth-scaled initialization method (Zhang et al., 2019) to handle gradient vanishing and

16. This differs from our previous finding (Zhang et al., 2018a) where the decoding time of Transformer grows near exponentially with the increase of source sentence length due to the lack of the cache implementation.

Activation	**Model**	**#Params**	**BLEU**	**METEOR**
Transformer		72.31M	27.59	47.93
exp	`AAN+` w/ Ner	72.30M	27.56	47.81
	`AAN+` w/ Wet	73.88M	27.49	**48.01**
Relu	`AAN+` w/ Ner	72.30M	27.50	47.75
	`AAN+` w/ Wet†	73.88M	17.38	37.88
Square	`AAN+` w/ Ner	72.30M	27.35	47.64
	`AAN+` w/ Wet†	73.88M	27.54	47.78

Table 5: Case-sensitive tokenized BLEU and METEOR for different activation functions on WMT14 En-De (newstest2014). *Relu*: $\text{Relu}(z) = \max\{z, 0\}$; *Square*: $\text{Square}(z) = z^2$. †: we apply gradient norm clipping of value 1.0 to stabilize the training. Note that Far is not well-defined with these activations.

train a 12-layer Transformer base model. For big Transformer, we increase the model dimension, the number of attention heads and the FFN size to 1024, 16 and 4096, respectively, and we set the warmup step size to 16K and ($\beta_1 = 0.9$, $\beta_2 = 0.998$) for Adam.

Table 4 summarizes the translation results on WMT14 En-De. Regardless of evaluation metrics and model settings, `AAN+` yields competitive performance compared to the baseline. Ner performs the best among different patterns with respect to BLEU, slightly surpassing the baseline by 0.08 in the deep setting. All `AAN+` models accelerate decoding, and enlarges the decoding speedup from $1.16\times \sim 1.32\times$ to $1.22\times \sim 1.40\times$ (base$\rightarrow$deep/big). Note Wet performs better in the big setting in inference speed, which based on our analysis is caused by producing significantly shorter translations. By contrast, the acceleration of Linear Attention deteriorates when scaling from base to deep and big; SSRU performs slightly worse in the big setting. In short, `AAN+` generalizes to larger Transformer models with higher decoding speedup, showing its superiority.

6.3.7 Alternatives to *exp*$(\cdot)$

Following the dot-product attention (Vaswani et al., 2017), we employ the exponential activation function $\exp(\cdot)$ to produce attention weights for `AAN+`, as shown in Equation (8-10). This function satisfies two constraints: 1) its output is positive, ensuring that its normalization follows the categorical distribution; and 2) it is a monotone function such that Ner and Far are well-defined. In this section, we study its alternatives (*Relu* and *Square*) by relaxing the second constraint.

Table 5 shows the results. We observe that neither Relu nor Square beats the original exp function except for Wet with Square which achieves a gain of 0.05 BLEU. Unfortunately, Wet with these alternatives becomes very unstable during training, relying on gradient normalization to converge. Thus, it's still favorable to use *exp* as the activation function, which offers flexibility, ensures training stability and delivers promising translation quality.

Settings				#Params	△Decode	BLEU	METEOR
Avg	Far	Ner	Wet				
Transformer Baseline				72.31M	1.00×	27.59	47.93
✓				72.30M	1.24×	27.50	47.80
	✓			72.30M	1.16×	27.35	47.73
		✓		72.30M	1.15×	27.56	47.81
			✓	73.88M	1.13×	27.49	**48.01**
	✓	✓		72.30M	**1.21**×	27.71	47.92
✓		✓		72.30M	1.24×	27.34	47.81
		✓	✓	73.88M	1.11×	**27.77**	47.89
	✓	✓	✓	73.88M	1.13×	27.27	47.65
✓	✓	✓		72.30M	1.20×	27.46	47.76
✓	✓	✓	✓	73.88M	1.10×	27.49	47.68

Table 6: Case-sensitive tokenized BLEU and METEOR on WMT14 En-De (newstest2014). The patterns are complementary to some degree.

6.3.8 Results on Pattern Complementarity

Different patterns endow AAN+ with different inductive biases in handling the contextual information. We next analyze whether their combination could result in better translation performance. We feed the input to different AAN+ models and then adopt an average pooling layer to get the final output.

Table 6 shows that most combinations, unfortunately, hurt the translation quality. Diverse contexts are not always complementary, which might result in negative interference degenerating token representations. However, one exception we notice is to inject the neighboring information into the weighting schema (Wet+Ner) or the distant schema (Far+Ner), which yield a BLEU score of 27.71 and 27.77, surpassing the baseline by 0.12 and 0.18, respectively, albeit with lower METEOR and smaller decoding speedup. This suggests that different patterns are complementary with a varying degree.

6.4 Results on Other Translation tasks

We also report results on other translation tasks. Table 7 and Table 8 list the translation results on WMT14 En-Fr and NIST Zh-En, respectively. Our baseline achieves a BLEU score of 39.09 on WMT14 En-Fr, surpassing the value reported in (Vaswani et al., 2017) by around 1.0 BLEU. Overall, AAN+ performs competitively against Transformer across different language pairs in terms of BLEU and METEOR, with a decoding speedup of 1.15× $\sim$ 1.31×. This demonstrates that AAN+ generalizes to different language pairs and training data sizes.

On some language pairs, AAN+ even surpasses Transformer: +1.05/+0.9 BLEU/METEOR on WMT14 En-Fr and +0.28 BLEU on NIST Zh-En. Compared to SSRU, AAN+ delivers sim-

Model	**#Params**	**△Train**	**△Decode**	**BLEU**	**METEOR**
Transformer	76.34M	1.00×	1.00×	39.09	60.58
SSRU	73.18M	0.82×	1.33×	38.74	60.56
Linear Attention	76.34M	0.51×	0.23×	39.40	60.95
AAN+ w/ Avg	76.33M	**1.03**×	1.30×	38.61	60.46
AAN+ w/ Far	76.33M	1.01×	1.30×	38.06	60.16
AAN+ w/ Ner	76.33M	1.01×	**1.31**×	**40.14**	**61.48**
AAN+ w/ Wet	77.90M	0.97×	1.19×	38.46	60.35

Table 7: Case-sensitive tokenized BLEU and METEOR for different models on WMT14 En-Fr. The decoding time is obtained on the test set and averaged over 5 different runs.

Model	**#Params**	**△Train**	**△Decode**	**TAVG**
Transformer	69.26M	1.00×	1.00×	46.23 (**34.75**)
SSRU	66.10M	0.86×	1.32×	45.80 (34.45)
Linear Attention	69.26M	0.49×	0.24×	46.05 (34.70)
AAN+ w/ Avg	69.25M	**1.04**×	1.28×	46.27 (34.55)
AAN+ w/ Far	69.25M	1.03×	1.27×	45.77 (34.07)
AAN+ w/ Ner	69.25M	1.03×	**1.28**×	46.38 (34.52)
AAN+ w/ Wet	70.83M	0.98×	1.15×	**46.51** (34.67)

Table 8: Case-sensitive tokenized BLEU (METEOR) on NIST Zh-En. *TAVG* denotes the average score over NIST02, NIST03, NIST04, NIST06 and NIST08. The decoding time is collected on all test sets with a batch size of 32.

ilar decoding speeds but with significant quality improvement; AAN+ also outperforms Linear Attention in both translation performance and running efficiency. In terms of BLEU, Ner shows the best generalization ability and decoding efficiency among all patterns, though Wet obtains the best quality on NIST Zh-En. These results further confirm the advantage of AAN+ (Ner in particular) in accelerating decoding and also its capability of handling diverse linguistic and syntactic structures as well as different data conditions.

6.5 Results on Table-to-Text Generation

Different from machine translation, table-to-text generation requires to deal with the relation of different entries in the table-style data (Liu et al., 2018). Table 9 shows the ability of Transformer in handling sequential table data, surpassing previous structural model (Liu et al., 2018) by 4.43 ROUGE. Compared to Transformer, AAN+ achieves comparable performance with marginal improvement when using Ner (+0.16 BLEU and +0.08 ROUGE), and yields a decoding speedup of $1.02\times \sim 1.26\times$. The average pattern Avg operates fastest

Model	**#Params**	**△Train**	**△Decode**	**BLEU-4**	**R-4**
Structure-aware Seq2Seq*	-	-	-	44.89	41.65
Transformer	67.56M	1.00×	1.00×	44.67	46.08
SSRU	64.41M	0.68×	1.15×	44.84	45.82
Linear Attention	67.56M	0.64×	0.60×	44.80	46.14
AAN+ w/ Avg	67.56M	**1.03**×	**1.26**×	44.41	45.88
AAN+ w/ Far	67.56M	1.03×	1.06×	44.10	45.89
AAN+ w/ Ner	67.56M	1.03×	1.10×	**44.83**	**46.16**
AAN+ w/ Wet	69.13M	1.00×	1.02×	44.63	46.06

Table 9: BLEU-4 and ROUGE-4 (R-4) of different models on *WIKIBIO*. *: results reported by Liu et al. (2018). Decoding time is measured with a batch size of 10.

Model	**#Params**	**△Train**	**△Decode**	**R-1**	**R-2**	**R-L**
Pointer-Generator†	-	-	-	39.12	17.35	36.12
CopyTransformer†	-	-	-	39.25	17.54	36.45
Bottom-Up†	-	-	-	41.22	18.68	38.34
Transformer	60.31M	1.00×	1.00×	**39.50**	**17.97**	**36.88**
SSRU	57.15M	0.54×	1.19×	36.27	16.09	33.81
Linear Attention	60.31M	0.79×	0.75×	38.35	17.22	35.77
AAN+ w/ Avg	60.30M	**1.02**×	1.48×	37.44	16.72	34.93
AAN+ w/ Far	60.30M	1.01×	**1.50**×	36.18	16.31	33.90
AAN+ w/ Ner	60.30M	1.01×	1.26×	37.58	16.74	35.01
AAN+ w/ Wet	61.88M	1.00×	1.25×	37.69	16.76	35.19

Table 10: ROUGE scores (F-Measure) of neural abstractive document summarization models on *CNN/Daily Mail*. †: results reported by Gehrmann et al. (2018). *R-1*, *R-2* and *R-L* denote ROUGE-1, ROUGE-2, and ROUGE-L, respectively. Decoding time is measured with a batch size of 10.

(1.03× for training and 1.26 × for decoding) outperforming SSRU and Linear Attention, albeit slightly underperforming Transformer in BLEU (-0.26) and ROUGE (0.20). AAN+ is compatible with structural inputs.

6.6 Results on Document Summarization

Compared with other generation tasks, document summarization stands out with its lengthy target sequence, which offers an ideal testbed to examine the model's ability of handling long-range dependency. Table 10 summarizes the results for different models.

Reference	david lynch says he wo n't be directing new episodes of twin peaks . showtime " saddened " over decision , which involved a dispute over money .
Transformer	director david lynch has confirmed he will no longer direct the revival of " twin peaks " . the offbeat tv series was set to return in 2016 . the groundbreaking series is considered one of the most influential shows in television history .
`AAN+` *w/* Avg	film director david lynch has confirmed he will no longer direct the revival of ' twin peaks ' . the offbeat tv series was set to return in 2016 . lynch broke the news about his departure in a series of tweets .
`AAN+` *w/* Far	david lynch has confirmed he will no longer direct the revival of ' twin peaks ' . the series is considered one of the most influential tv shows in tv history . lynch broke the news about his departure in a series of tweets .
`AAN+` *w/* Ner	david lynch confirms he will no longer direct the revival of ' twin peaks ' . the offbeat tv series , created by lynch and mark frost , featured a quirky fbi agent . the groundbreaking series is considered one of the most influential shows in television history .
`AAN+` *w/* Wet	david lynch has confirmed he will no longer direct the revival of ' twin peaks ' . the series is considered one of the most influential tv shows in tv history . lynch broke the news about his departure in a series of tweets .

Table 11: Case study on document summarization. Although Transformer outperforms `AAN+` in ROUGE, we observe no significant difference from the model outputs.

We didn't apply the pointer-generator method and the coverage penalty (Gehrmann et al., 2018), but our Transformer baseline matches the performance of existing models[17]. We observe that `AAN+` achieves significantly better decoding acceleration, 1.48×/1.25× faster with Avg/Wet at the cost of 1.95/1.69 ROUGE-L, and that Ner outperforms Avg in ROUGE. Although SSRU performs well on translation tasks, it generalizes poorly to summarization, yielding worse quality and decoding speedup compared to `AAN+`. By contrast, Linear Attention outperforms `AAN+` in ROUGE and shows better decoding efficiency but it's still significantly slower than `AAN+`.

We also show a case study in Table 11. Unlike translation or table-to-text generation, document summarization lacks a strictly defined semantic correspondence between the input document and its summary. Thus, it is more difficult for abstractive neural models to generate hypotheses that match the given reference very well. In practice, we find that neural models prefer to copy sentences near the beginning of the document, while fails to summarize information over the whole passage. Results in Table 11 suggest that `AAN+` is

17. We find that Transformer can automatically handle the copying and repeation problem with subwords.

capable of generating reasonable summarizations that are on par with those of Transformer, although AAN+ prefers generating relatively shorter sequences.

7. Conclusion

In this paper, we have presented the generalized average attention network (AAN+), which formulates average attention network (Zhang et al., 2018a) in a more principle way that enables us to explore richer heuristic attention patterns for generation, including position-based patterns and content-based patterns. We studied four attention patterns inspired by the analysis on the learned self-attention weights in the Transformer decoder, among which the neighboring pattern (Ner) generalizes the best over different tasks while performs on par with the average pattern in decoding efficiency. Different patterns show complementarity to a varying degree.

Our work shows the feasibility of simplifying the decoder SAN, as well as the importance of local context to generation. Extensive experiments on machine translation with three language pairs, table-to-text generation and document summarization demonstrate the superiority of AAN+ in accelerating decoding with little or no loss in generation quality (except for summarization). Our analysis on long sentence generation reveals that attentions defined by AAN+ are capable of handling long-range dependency, albeit being slightly weaker than SAN.

Acknowledgements

We thank the reviewers for their insightful comments. The project was supported by National Key Research and Development Program of China (No. 2020AAA0108004), National Natural Science Foundation of China (No. 62276219), Natural Science Foundation of Fujian Province of China (No. 2020J06001), Youth Innovation Fund of Xiamen (No. 3502Z20206059), and Key Research and Development Program of Yunnan Province (Grant No. 202203AA080004-2).

References

Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of the International Conference on Learning Representations (ICLR)*, Vol. abs/1409.0473.

Barrault, L., Bojar, O., Costa-jussà, M. R., Federmann, C., Fishel, M., Graham, Y., Haddow, B., Huck, M., Koehn, P., Malmasi, S., Monz, C., Müller, M., Pal, S., Post, M., & Zampieri, M. (2019). Findings of the 2019 conference on machine translation (WMT19). In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pp. 1–61, Florence, Italy. Association for Computational Linguistics.

Beltagy, I., Peters, M. E., & Cohan, A. (2020). Longformer: The long-document transformer..

Bojar, O., Buck, C., Federmann, C., Haddow, B., Koehn, P., Leveling, J., Monz, C., Pecina, P., Post, M., Saint-Amand, H., et al. (2014). Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the ninth workshop on statistical machine translation*, pp. 12–58.

Chelba, C., Chen, M., Bapna, A., & Shazeer, N. (2020). Faster transformer decoding: N-gram masked self-attention..

Choromanski, K. M., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J. Q., Mohiuddin, A., Kaiser, L., Belanger, D. B., Colwell, L. J., & Weller, A. (2021). Rethinking attention with performers. In *International Conference on Learning Representations*.

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*.

Denkowski, M., & Lavie, A. (2014). Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*.

Gehring, J., Auli, M., Grangier, D., & Dauphin, Y. (2017a). A convolutional encoder model for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 123–135, Vancouver, Canada. Association for Computational Linguistics.

Gehring, J., Auli, M., Grangier, D., Yarats, D., & Dauphin, Y. N. (2017b). Convolutional sequence to sequence learning. In Precup, D., & Teh, Y. W. (Eds.), *Proceedings of the 34th International Conference on Machine Learning*, Vol. 70 of *Proceedings of Machine Learning Research*, pp. 1243–1252, International Convention Centre, Sydney, Australia. PMLR.

Gehrmann, S., Deng, Y., & Rush, A. (2018). Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 4098–4109.

Gu, J., Bradbury, J., Xiong, C., Li, V. O., & Socher, R. (2018). Non-autoregressive neural machine translation. In *International Conference on Learning Representations*.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep residual learning for image recognition. *CoRR*, *abs/1512.03385*.

Hermann, K. M., Kocisky, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., & Blunsom, P. (2015). Teaching machines to read and comprehend. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., & Garnett, R. (Eds.), *Advances in Neural Information Processing Systems 28*, pp. 1693–1701. Curran Associates, Inc.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, *9*, 1735–1780.

Junczys-Dowmunt, M., Heafield, K., Hoang, H., Grundkiewicz, R., & Aue, A. (2018). Marian: Cost-effective high-quality neural machine translation in C++. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pp. 129–135, Melbourne, Australia. Association for Computational Linguistics.

Kasai, J., Pappas, N., Peng, H., Cross, J., & Smith, N. (2021). Deep encoder, shallow decoder: Reevaluating non-autoregressive machine translation. In *International Conference on Learning Representations*.

Katharopoulos, A., Vyas, A., Pappas, N., & Fleuret, F. (2020). Transformers are RNNs: Fast autoregressive transformers with linear attention. In III, H. D., & Singh, A. (Eds.), *Proceedings of the 37th International Conference on Machine Learning*, Vol. 119 of *Proceedings of Machine Learning Research*, pp. 5156–5165. PMLR.

Kim, Y., Denton, C., Hoang, L., & Rush, A. M. (2017). Structured attention networks. In *International Conference on Learning Representations*.

Kim, Y. J., Junczys-Dowmunt, M., Hassan, H., Fikri Aji, A., Heafield, K., Grundkiewicz, R., & Bogoychev, N. (2019). From research to production and back: Ludicrously fast neural machine translation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pp. 280–288, Hong Kong. Association for Computational Linguistics.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In Bengio, Y., & LeCun, Y. (Eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Kitaev, N., Kaiser, L., & Levskaya, A. (2020). Reformer: The efficient transformer. In *International Conference on Learning Representations*.

Kobayashi, G., Kuribayashi, T., Yokoi, S., & Inui, K. (2020). Attention is not only a weight: Analyzing transformers with vector norms. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7057–7075, Online. Association for Computational Linguistics.

Lebret, R., Grangier, D., & Auli, M. (2016). Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1203–1213, Austin, Texas. Association for Computational Linguistics.

Lee, K., Levy, O., & Zettlemoyer, L. (2017). Recurrent additive networks. *CoRR*, *abs/1705.07393*.

Lei, T., Zhang, Y., Wang, S. I., Dai, H., & Artzi, Y. (2018). Simple recurrent units for highly parallelizable recurrence. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 4470–4481, Brussels, Belgium. Association for Computational Linguistics.

Liu, L., Utiyama, M., Finch, A., & Sumita, E. (2016). Neural machine translation with supervised attention. In *Proc. of COLING 2016*, pp. 3093–3102, Osaka, Japan. The COLING 2016 Organizing Committee.

Liu, T., Wang, K., Sha, L., Chang, B., & Sui, Z. (2018). Table-to-text generation by structure-aware seq2seq learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Luong, T., Pham, H., & Manning, C. D. (2015). 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. Association for Computational Linguistics.

Mi, H., Wang, Z., & Ittycheriah, A. (2016). Supervised attentions for neural machine translation. In *Proc. of EMNLP*, pp. 2283–2288, Austin, Texas. Association for Computational Linguistics.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., & Weinberger, K. Q. (Eds.), *Advances in Neural Information Processing Systems 26*, pp. 3111–3119. Curran Associates, Inc.

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, ACL '02, pp. 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Peng, H., Pappas, N., Yogatama, D., Schwartz, R., Smith, N., & Kong, L. (2021). Random feature attention. In *International Conference on Learning Representations*.

Raganato, A., Scherrer, Y., & Tiedemann, J. (2020). Fixed encoder self-attention patterns in transformer-based machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 556–568, Online. Association for Computational Linguistics.

Sennrich, R., Haddow, B., & Birch, A. (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)*, pp. 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Shen, T., Jiang, J., Zhou, T., Pan, S., Long, G., & Zhang, C. (2018). Disan: Directional self-attention network for rnn/cnn-free language understanding. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press.

Srivastava, R. K., Greff, K., & Schmidhuber, J. (2015). Highway networks. *CoRR*, *abs/1505.00387*.

Stern, M., Shazeer, N., & Uszkoreit, J. (2018). Blockwise parallel decoding for deep autoregressive models. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., & Garnett, R. (Eds.), *Advances in Neural Information Processing Systems 31*, pp. 10086–10095. Curran Associates, Inc.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., & Weinberger, K. Q. (Eds.), *Advances in Neural Information Processing Systems 27*, pp. 3104–3112. Curran Associates, Inc.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., & Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., & Garnett, R. (Eds.), *Advances in Neural Information Processing Systems 30*, pp. 5998–6008. Curran Associates, Inc.

Voita, E., Talbot, D., Moiseev, F., Sennrich, R., & Titov, I. (2019). Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 5797–5808, Florence, Italy. Association for Computational Linguistics.

Wang, C., Zhang, J., & Chen, H. (2018). Semi-autoregressive neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 479–488, Brussels, Belgium. Association for Computational Linguistics.

Yang, B., Tu, Z., Wong, D. F., Meng, F., Chao, L. S., & Zhang, T. (2018a). Modeling localness for self-attention networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 4449–4458, Brussels, Belgium. Association for Computational Linguistics.

Yang, J., Zhang, B., Qin, Y., Zhang, X., Lin, Q., & Su, J. (2018b). Otem&utem: Over-and under-translation evaluation metric for nmt. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pp. 291–302. Springer.

You, W., Sun, S., & Iyyer, M. (2020). Hard-coded Gaussian attention for neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7689–7700, Online. Association for Computational Linguistics.

Zhang, B., Xiong, D., & Su, J. (2020). Neural machine translation with deep attention. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, *42*(01), 154–163.

Zhang, B., & Sennrich, R. (2019). A lightweight recurrent network for sequence modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 1538–1548, Florence, Italy. Association for Computational Linguistics.

Zhang, B., Su, J., Xiong, D., Lu, Y., Duan, H., & Yao, J. (2015). Shallow convolutional neural network for implicit discourse relation recognition. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 2230–2235.

Zhang, B., Titov, I., & Sennrich, R. (2019). Improving deep transformer with depth-scaled initialization and merged attention. 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)*, pp. 898–909, Hong Kong, China. Association for Computational Linguistics.

Zhang, B., Titov, I., & Sennrich, R. (2020). Fast interleaved bidirectional sequence generation. In *Proceedings of the Fifth Conference on Machine Translation*, pp. 503–515, Online. Association for Computational Linguistics.

Zhang, B., Xiong, D., & Su, J. (2018a). Accelerating neural transformer via an average attention network. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1789–1798, Melbourne, Australia. Association for Computational Linguistics.

Zhang, B., Xiong, D., Su, J., Lin, Q., & Zhang, H. (2018b). Simplifying neural machine translation with addition-subtraction twin-gated recurrent networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 4273–4283, Brussels, Belgium. Association for Computational Linguistics.

Zhang, B., Xiong, D., Xie, J., & Su, J. (2020). Neural machine translation with gru-gated attention model. *IEEE Transactions on Neural Networks and Learning Systems*, *31*(11), 4688–4698.

Zhang, H., Gong, Y., Shen, Y., Li, W., Lv, J., Duan, N., & Chen, W. (2021). Poolingformer: Long document modeling with pooling attention. In *International Conference on Machine Learning*, pp. 12437–12446. PMLR.