# An Empirical Study of Adequate Vision Span for Attention-Based Neural Machine Translation

**Raphael Shu, Hideki Nakayama**
shu@nlab.ci.i.u-tokyo.ac.jp, nakayama@ci.i.u-tokyo.ac.jp
The University of Tokyo

## Abstract

Recently, the attention mechanism plays a key role to achieve high performance for Neural Machine Translation models. However, as it computes a score function for the encoder states in all positions at each decoding step, the attention model greatly increases the computational complexity. In this paper, we investigate the adequate vision span of attention models in the context of machine translation, by proposing a novel attention framework that is capable of reducing redundant score computation dynamically. The term "vision span" means a window of the encoder states considered by the attention model in one step. In our experiments, we found that the average window size of vision span can be reduced by over 50% with modest loss in accuracy on English-Japanese and German-English translation tasks.

## 1 Introduction

In recent years, recurrent neural networks have been successfully applied in machine translation. In many major language pairs, Neural Machine Translation (NMT) has already outperformed conventional Statistical Machine Translation (SMT) models (Luong et al., 2015b; Wu et al., 2016).

NMT models are generally composed of an encoder and a decoder, which is also known as encoder-decoder framework (Sutskever et al., 2014). The encoder creates a vector representation of the input sentence, whereas the decoder generates the translation from this single vector. This simple encoder-decoder model suffers from a long backpropagation path; thus, adversely affected by long input sequences.

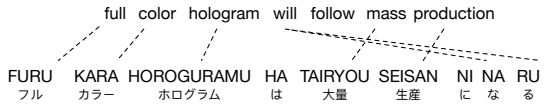
In recent NMT models, soft attention mechanism (Bahdanau et al., 2014) has been a key extension to ensure high performance. In each decoding step, the attention model computes alignment weights for all the encoder states. Then a context vector, which is a weighted summarization of the encoder states is computed and fed into the decoder as input. In contrast to the aforementioned simple encoder-decoder model, the attention mechanism can greatly shorten the backpropagation path.

Although the attention mechanism provides NMT models with a boost in performance, it also significantly increases the computational burden. As the attention model has to compute the alignment weights for all the encoder states in each step, the decoding process becomes time-consuming. Even worse, recent researches in NMT prefer to separate the texts into subwords (Sennrich et al., 2016) or even characters (Chung et al., 2016), which means massive encoder states have to be considered in the attention model at each step, thereby resulting in increasing computational cost. On the other hand, the attention mechanism is becoming more complicated. For example, the NMT model with recurrent attention modeling (Yang et al., 2016) maintains a dynamic memory of attentions for every encoder states, which is updated in each decoding step.

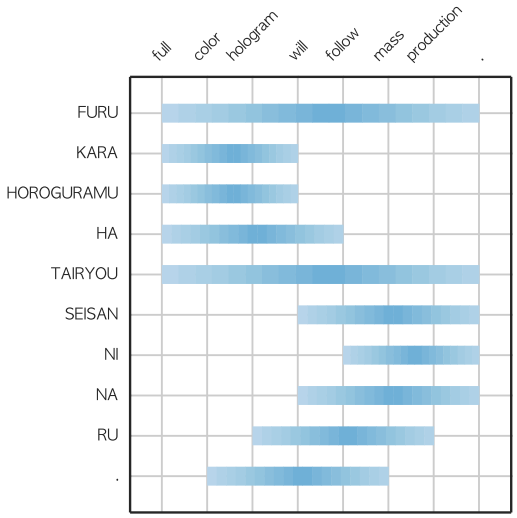
In this paper, we study the adequate *vision span* in the context of machine translation. Here, the term "vision span" means a window of encoder states considered by the attention model in one step. We examine the minimum window size of an attention model have to consider in each step while maintaining the translation quality. For this purpose, we propose a novel attention framework which we refer to as Flexible Attention in this paper. The proposed attention framework tracks the center of attention in each decoding step, and pre-



*Proceedings of the First Workshop on Neural Machine Translation*, pages 1–10,
Vancouver, Canada, August 4, 2017. 




(a)




(b)

Figure 1: (a) An example English-Japanese sentence pair with long-range reordering (b) The vision span predicted by the proposed Flexible Attention at each step in English-Japanese translation task

dict an adequate vision span for the next step. In the test time, the encoder states outside of this range are omitted in the computation of score function.

Our proposed attention framework is based on simple intuition. For most language pairs, the translations of words inside a phrase usually remain together. Even the translation of a small chunk usually does not mix with the translation of other words. Hence, information about distant words is basically unnecessary when translating locally. Therefore, we argue that computing the attention over all positions in each step is redundant. However, attending to distant positions remains important when dealing with long-range reordering. In Figure 1(a), we show an example sentence pair with long-range reordering, where the positions of the first three words have monotone alignments, but the fourth word is aligned to distant target positions. If we can predict whether the next word to translate is in a local position, the amount of redundant computation in the attention model can be safely reduced by controlling the window size of vision span dynamically. This motivated us to propose a flexible attention framework which predicts the minimum required vision span according to the context (See Figure 1(b)).

We evaluated our proposed Flexible Attention by comparing with the conventional attention mechanism, and Local Attention (Luong et al., 2015a) which puts attention on a fixed-size window. We focus on comparing the minimum window size of vision span these models can achieve without hurting the performance too much. Note that as the window size determines the number of times the score function is evaluated, reducing the window size leads to the reduction of score computation. We select English-Japanese and German-English language pairs for evaluation as they consi st of languages with different word orders, which means the attention model cannot simply look at a local range constantly and translate monotonically. Through empirical evaluation, we found with Flexible Attention, the average window size is reduced by 56% for English-Japanese task and 64% for German-English task, with modest loss of accuracy. The reduction rate also achieves 46% for character-based NMT models.

Our contributions can be summarized as three folds:

1. We empirically confirmed that the conventional attention mechanism performs a significant amount of redundant computation. Although attending globally is necessary when dealing with long-range reordering, a small vision span is sufficient when translating locally. The results may provide insights for future research on more efficient attention-based NMT models.

2. The proposed Flexible Attention provides a general framework for reducing the amount of score computation according to the context, which can be combined with other expensive attention models of which computing for all positions in each step is costly.

3. We found that reducing the amount of computation in the attention model can benefit the decoding speed on CPU, but not GPU.

## 2 Attention Mechanism in NMT

Although the network architectures of NMT models differ in various respects, they generally fol-

low the encoder-decoder framework. In Bahdanau et al. (2014), a bidirectional recurrent neural network is used as the encoder, which accepts the embeddings of input words. The hidden states $\bar{\boldsymbol{h}}_{\boldsymbol{1}}, ..., \bar{\boldsymbol{h}}_{\boldsymbol{S}}$ of the encoder are then used in the decoding phase. Basically, the decoder is composed of a recurrent neural network (RNN). The decoder RNN computes the next state based on the embedding of the previously generated word, and a context vector given by the attention mechanism. Finally, the probabilities of output words in each time step are predicted based on the decoder states $\boldsymbol{h}_{\boldsymbol{1}}, ..., \boldsymbol{h}_{\boldsymbol{N}}$.

The soft attention mechanism (Karol et al., 2015) is introduced to NMT in Bahdanau et al. (2014), which computes a weighted summarization of all encoder states in each decoding step, to obtain the context vector:

$$\boldsymbol{c}_t = \sum_s a_t(s) \bar{\boldsymbol{h}}_{\boldsymbol{s}} \,, \tag{1}$$

where $\bar{\boldsymbol{h}}_{\boldsymbol{s}}$ is the $s$-th encoder state, $a_t(s)$ is the alignment weight of $\bar{\boldsymbol{h}}_{\boldsymbol{s}}$ in decoding step $t$. The calculation of $a_t(s)$ is given by the softmax of the weight scores:

$$a_t(s) = \frac{\exp(\text{score}(\boldsymbol{h}_{\boldsymbol{t-1}}, \bar{\boldsymbol{h}}_{\boldsymbol{s}}))}{\sum_{s'} \exp(\text{score}(\boldsymbol{h}_{\boldsymbol{t-1}}, \bar{\boldsymbol{h}}_{\boldsymbol{s'}}))} \,. \tag{2}$$

The unnormalized weight scores are computed with a score function, defined as[1]:

$$\text{score}(\boldsymbol{h}_{\boldsymbol{t-1}}, \bar{\boldsymbol{h}}_{\boldsymbol{s}}) = \boldsymbol{v}_{\boldsymbol{a}}^{\top} \tanh(\boldsymbol{W}_{\boldsymbol{a}}[\boldsymbol{h}_{\boldsymbol{t-1}}; \bar{\boldsymbol{h}}_{\boldsymbol{s}}]) \,, \tag{3}$$

where $\boldsymbol{v}_{\boldsymbol{a}}$ and $\boldsymbol{W}_{\boldsymbol{a}}$ are the parameters of the score function, $[\boldsymbol{h}_{\boldsymbol{t-1}}; \bar{\boldsymbol{h}}_{\boldsymbol{s}}]$ is a concatenation of the decoder state in the previous step and an encoder state. Intuitively, the alignment weight indicates whether an encoder state is valuable for generating the next output word. Note that many discussions on alternative ways for computing the score function can be found in Luong et al. (2015a).

## 3 Flexible Attention

In this section, we present our main idea for reducing the window size of vision span. In contrast to conventional attention models, we track the center of attention in each decoding step with

$$p_t = \sum_s a_t(s) \cdot s \,. \tag{4}$$

The value of $p_t$ provides an approximate focus of attention in time step $t$. Then we penalize the alignment weights for the encoder states distant from $p_{t-1}$, which is the focus in the previous step. This is achieved by a position-based penalty function:

$$\text{penalty}(s) = g(t) d(s, p_{t-1}) \,, \tag{5}$$

where $g(t)$ is a *sigmoid* function that adjusts the strength of the penalty dynamically based on the context in step $t$. $d(s, p_{t-1})$ provides the distance between position $s$ and the previous focus $p_{t-1}$, which is defined as:

$$d(s, p_{t-1}) = \frac{1}{2\sigma^2}(s - p_{t-1})^2 \,. \tag{6}$$

Hence, distant positions attract exponentially large penalties. The denominator $2\sigma^2$, which is a hyperparameter, controls the maximum of penalty when $g(t)$ outputs 1.

The position-based penalty function is finally integrated into the computation of the alignment weights as:

$$a_t(s) = \frac{\exp(\text{score}(\boldsymbol{h}_{\boldsymbol{t-1}}, \bar{\boldsymbol{h}}_{\boldsymbol{s}}) - \text{penalty}(s))}{\sum_{s'} \exp(\text{score}(\boldsymbol{h}_{\boldsymbol{t-1}}, \bar{\boldsymbol{h}}_{\boldsymbol{s'}}) - \text{penalty}(s'))}, \tag{7}$$

where the penalty function acts as a second score function that penalize encoder states only by their positions. When $g(t)$ outputs zero, the penalty function will have no effects on alignment weights. Note that the use of distance-based penalties here is similar in appearance to Local Attention (local-p) proposed in Luong et al. (2015a). The difference is that Local Attention predicts the center of attention in each step and attends to a fixed window. Further discussion will be given later in Section 4.

In this paper, the strength function $g(t)$ in Equation 5 is defined as:

$$g(t) = \text{sigmoid}(\boldsymbol{v}_{\boldsymbol{g}}^{\top} \tanh(\boldsymbol{W}_{\boldsymbol{g}}[\boldsymbol{h}_{\boldsymbol{t-1}}; \boldsymbol{i}_{\boldsymbol{t}}]) + b_g), \tag{8}$$

where $\boldsymbol{v}_{\boldsymbol{g}}$, $\boldsymbol{W}_{\boldsymbol{g}}$ and $b_g$ are parameters. We refer to this attention framework as Flexible Attention in

[1] In the original paper (Bahdanau et al., 2014), the equation of the score function is a sum. Here, we use a concatenation in Equation 3 in order to align with (Luong et al., 2015a), which is an equivalent form of the original equation.

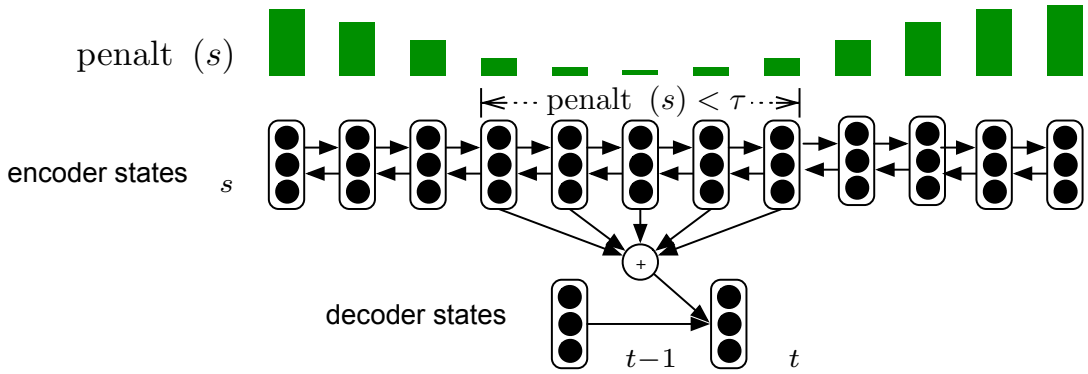



Figure 2: Illustration of the way that Flexible Attention reduces the window of vision span. In each decoding step, only a portion of the encoder states are selected by the position-based penalty function to compute the alignment weights.

this paper, as the window of effective attention is adjusted by $g(t)$ in according to the context.

Intuitively, when the model is translating inside a phrase, the alignment weights for distant positions can be safely penalized by letting $g(t)$ output a high value. If the next word is expected to be translated from a new phrase, $g(t)$ shall output a low value to allow attending to any position. Actually, the selected output word in the previous step can greatly influence this decision, as selection of the output word can determine whether the translation of a phrase is complete. Therefore, the embedding of the feedback word $\boldsymbol{i_t}$ is put into the equation.

### 3.1 Reducing Window Size of Vision Span

As we can see from Equation 7, if a position is heavily penalized, then it will be assigned a low attention probability regardless of the value of the score function. In the test time, we can set a threshold $\tau$, and only compute the score function for positions with penalties lower than $\tau$. Figure 2 provides an illustration of the selection process. The selected range can be obtained by solving $penalty(s) < \tau$, which gives $s \in \left( p_{t-1} - \sigma\sqrt{2\tau/g(t)},\ p_{t-1} + \sigma\sqrt{2\tau/g(t)} \right)$.

Because the strength term $g(t)$ in Equation 5 only needs to be computed once in each step, the computational cost of the penalty function does not increase as the input length increases. By utilizing the penalty values to omit computation of the score function, the totally computational cost can be reduced.

Although a low threshold would lead to further reduction of the window size of vision span, the performance degrades as information from the source side will be greatly limited. In practice, we can find a good threshold to balance the tradeoff of performance and computational cost on a validation dataset.

### 3.2 Fine-tuning for Better Performance

In order to further narrow down the vision span, we want $g(t)$ to output a large value to clearly differentiate valuable encoder states from other states encoder based on their positions. Thus, we can further fine-tune our model to encourage it to decode using larger penalties with the following loss function:

$$J = \sum_{i=1}^{D} -\log p(y^{(i)}|x^{(i)}) - \beta \frac{1}{T} \sum_{t=1}^{T} g(t)^{(i)}, \quad (9)$$

where $\beta$ is a hyperparameter to control the balance of cross-entropy and the average strength of penalty. In our experiments, we tested $\beta$ among $(0.1, 0.001, 0.0001)$ on a development data and found that setting $\beta$ to 0.1 and fine-tuning for one epoch works well. If we train the model with this loss function from the beginning, as the right part of the loss function is easier to be optimized, the value of $g(t)$ saturates quickly, which slows down the training process.

## 4 Related Work

To the best of our knowledge, only a limited number of related studies aimed to reduce the computational cost of the attention mechanism. Local Attention, which was proposed in Luong et al. (2015a), limited the range of attention to a fixed

window size. In Local Attention (local-p), the center of attention $p_t$ is *predicted* in each time step $t$:

$$p_t = S \cdot \text{sigmoid}(\boldsymbol{v_p}^\top \tanh(\boldsymbol{W_p h_t})) \,, \quad (10)$$

where $S$ is the length of the input sequence. Finally, the alignment weights are computed by:

$$\begin{aligned} a'_t(s) &= a_t(s) \exp(-\frac{(s-p_t)^2}{2\sigma^2}) \\ &= \frac{exp(\text{score}(\boldsymbol{h_{t-1}}, \bar{\boldsymbol{h}}_{\boldsymbol{s}}))}{\sum_{s'} exp(\text{score}(\boldsymbol{h_{t-1}}, \bar{\boldsymbol{h}}_{\boldsymbol{s'}}))} \exp(-\frac{(s-p_t)^2}{2\sigma^2}), \end{aligned} \quad (11)$$

where $\sigma$ is a hyperparameter determined by $\sigma = D/2$, where $D$ is a half of the window size. Local Attention only computes attention within the window $[p_t - D, p_t + D]$. In their work, the hyperparameter $D$ is empirically set to $D = 10$ for the English-German translation task, which means a window of 21 words.

Our proposed attention model differs from Local Attention in two key points: (1) our proposed attention model does not predict the fixiation of attention but tracks it in each step (2) the position-based penalty in our attention model is adjusted flexibly rather than remaining fixed. Note that in Equation 11 of Local Attention, the penalty term is applied outside the softmax function. In contrast, we integrate the penalty term with the score function (Eq. 7), such that the final probabilities still add up to 1.

Recently, a "cheap" linear model (de Brébisson and Vincent, 2016) is proposed to replace the attention mechanism with a low-complexity function. This cheap linear attention mechanism achieves an accuracy in the middle of Global Attention and a non-attention model on a question-answering dataset. This approach can be considered as another interesting way to balance the performance and computational complexity in sequence-generation tasks.

## 5 Experiments

In this section, we focus on evaluating our proposed attention models by measuring the minimum average window size of vision span it can achieve with a modest performance loss[2]. In detail, we measure the average number of the encoder states considered when computing the score function in Equation 3. Note that as we decode using Beam Search algorithm (Sutskever et al., 2014) , the value of window size is further averaged over the number of hypotheses considered in each step. For the conventional attention mechanism, as all positions have to be considered in each step, the average window size equals to the average sentence length of the testing data. Following Luong et al. (2015a), we refer to the conventional attention mechanism as Global Attention in experiments.

### 5.1 Experimental Settings

We evaluate our models on English-Japanese and German-English translation task. As translating these language pairs requires long-range reordering, the proposed Flexible Attention has to correctly predict when the reordering happens and look at distant positions when necessary. The training data of En-Ja task is based on ASPEC parallel corpus (Nakazawa et al., 2016), which contains 3M sentence pairs, whereas the test data contains 1812 sentences, which have 24.4 words on average. We select 1.5M sentence pairs according to the automatically calculated matching scores, which are provided along with the ASPEC corpus. For De-En task, we use the WMT'15 training data consisting of 4.5M sentence pairs. The WMT'15 test data (newstest2015) contains 2169 pairs, which have 20.7 words on average.

We preprocess the En-Ja corpus with "tokenizer.perl" for English side, and Kytea tokenizer (Neubig et al., 2011) for Japanese side. The preprocessing procedure for De-En corpus is similar to Li et al. (2014), except we did not filter sentence pairs with language detection.

The vocabulary size are cropped to 80k and 40k for En-Ja NMT models, whereas 50k for De-En NMT models. The OOV words are replaced with a "UNK" symbol. Long sentences with more than 50 words on either the source or target side are removed from the training set, resulting in 1.3M and 3.8M training pairs for En-Ja and De-En task respectively. We use mini-batch in our training procedure, where each batch contains 64 data samples. All sentence pairs are firstly sorted according to their length before we group them into batches. After which, the order of the mini-batches is shuffled.

[2]In our experiments, we try to limit the performance loss to be lower than 0.5 development BLEU. As the threshold $\tau$ is selected using a development corpus, the performance on test data is not ensured.

| Model | English-Japanese | | | German-English | |
|---|---|---|---|---|---|
| | window (words) | BLEU(%) | RIBES | window (words) | BLEU(%) |
| Global Attention baseline | 24.4 | 34.87 | 0.810 | 20.7 | 20.62 |
| Local Attention baseline | 18.4 | 34.52 | 0.809 | 15.7 | 21.09 |
| Flexible Attention ($\tau$=$\infty$) | 24.4 | 35.01 | 0.814 | 20.7 | 21.31 |
| Flexible Attention ($\tau$=1.2) | 16.4 | 34.90 | 0.812 | 7.8 | 21.11 |
| + fine-tuning ($\tau$=1.2) | **10.7** | 34.78 | 0.807 | **7.4** | 20.79 |

Table 1: Evaluation results on English-Japanese and German-English translation task. This table provides a comparison of the minimum window size of vision span the models can achieve with a modest loss of accuracy.

We adopt the network architecture described in Bahdanau et al. (2014) and set it as our baseline model. The size of word embeddings is 1000 for both languages. For the encoder, we use a bi-directional RNN composed of two LSTMs with 1000 units. For the decoder, we use a one-layer LSTM with 1000 units, where the input in each step is a concatenated vector of the embedding of the previous output $i_t$ and the context vector $c_t$ given by attention mechanism. Before the final softmax layer, we insert a fully-connected layer with 600 units to reduce the number of connections in the output layer.

For our proposed models, we empirically select $\sigma$ in Equation 6 from $(\frac{3}{2}, \frac{10}{2}, \frac{15}{2}, \frac{20}{2})$ on a development corpus. In our experiments, we found the attention models give the best trade-off between the window size and accuracy when $\sigma = 1.5$. Note that the value of $\sigma$ only determines the maximum of penalty when $g(t)$ outputs 1, but does not results in a fixed window size.

The NMT models are trained using Adam optimizer (Kingma and Ba, 2014) with an initial learning rate of 0.0001. We train the model for six epochs and start to halve the learning rate from the beginning of the fourth epoch. The maximum norm of the gradients is clipped to 3. Final parameters are selected by the smoothed BLEU (Lin and Och, 2004) on validation set. During test time, we use beam search with a beam size of 20.

In En-Ja task, we evaluate our implemented NMT models with BLEU and RIBES (Isozaki et al., 2010), in order to align with other researches on the same dataset. The results are reported following standard post-processing procedures[3]. For De-En task, we report *tokenized* BLEU [4].

### 5.2 Evaluations of Flexible Attention

We evaluate the attention models to determine the minimum window size they can achieve with a modest loss of accuracy (0.5 development BLEU) compared to Flexible Attention with $\tau = \infty$. The results we obtained are summarized in Table 1. The scores of Global Attention (conventional attention model) and Local Attention (Luong et al., 2015a) are listed for comparison. For Local Attention, we found a window size of 21 ($D = 10$) gives the best performance for En-Ja and De-En tasks. In this setting, Local Attention achieves an average window of 18.4 words in En-Ja task and 15.7 words in De-En task, as some sentences in the test corpus have fewer than 21 words.

For Flexible Attention, we search a good $\tau$ among $(0.8, 1.0, 1.2, 1.4, 1.6)$ on a development corpus so that the development BLEU(%) does not degrade more than 0.5 compared to $\tau = \infty$. Finally, $\tau = 1.2$ is selected for both language pairs in our experiments.

We can see from the results that Flexible Attention can achieve comparable scores even consider only half of the encoder states in each step. After fine-tuning, our proposed attention model further reduces 56% of the vision span for En-Ja task and 64% for De-En task. The high reduction rate confirms that the conventional attention model performs massive redundant computation. With Flexible Attention, redundant score computation can be efficiently cut down according to the context. Interestingly, the NMT models using Flexible Attention without the threshold improves the translation accuracy by a small margin, which may indi-

[3]We report the scores using Kytea tokenizer. The post-processing procedure for evaluation is described in http://lotus.kuee.kyoto-u.ac.jp/WAT/evaluation/

[4]The scores are produced by tokenizing with "tokenizer.perl" and evaluating with "multi-bleu.perl".

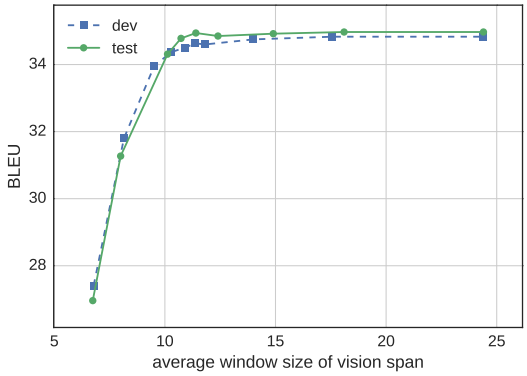



Figure 3: Trade off between window size and performance on the development and test data of English-Japanese tanslation task

cates that the quality of attention is improved.

### 5.3 Trade-off between Window Size and Accuracy

In order to figure out the relation between accuracy and the window size of vision span, we plot out the curve of the trade-off between BLEU score and average window size on En-Ja task, which is shown in Figure 3.

The data points are collected by testing different thresholds [5] with the fine-tuned Flexible Attention model. Interestingly, the NMT model with our proposed Flexible Attention suffers almost no loss in accuracy even the computations are reduced by half. Further trails to reduce the window size beneath 10 words will result in drastically degradation in performance.

### 5.4 Effects on Character-level Attention

| Model | window | BLEU | RIBES |
|---|---|---|---|
| Global Attention baseline | 144.9 | 26.18 | 0.767 |
| Flexible Attention ($\tau$=$\infty$) | 144.9 | 26.68 | 0.763 |
| Flexible Attention ($\tau$=1.0) | 80.4 | 26.18 | 0.757 |
| + fine-tuning ($\tau$=1.0) | **77.4** | 26.23 | 0.757 |

Table 2: Evaluation results with character-based English-Japanese NMT models

In order to examine the effects of Flexible Attention on extremely long character-level inputs, we also conducted experiments on character-based NMT models. We adopt the same network architecture as word-based models in the English-Japanese task, unless the sentences in both sides are tokenized into characters. We keep 100 most frequent types of character for the English side and 3000 types for the Japanese side. The embedding size is set to 100 for both sides. In order to train the models faster, all LSTMs in this experiment have 500 hidden units. The character-based models are trained 20 epochs with Adam optimizer with an initial learning rate of 0.0001. The learning rate begins to halve from 18-th epoch. After fine-tuning the model with the same hyperparameter ($\beta = 0.1$), we selected the threshold to be $\tau = 1.0$ in the same manner as the word-level experiment. We did not evaluate Local Attention in this experiment as selecting a proper fixed window size is time-consuming when the length of input sequence is extremely long.

The experimental results of character-based models are summarized in Table 2. Note that although the performance of character-based models can not compete with word-based model, the focus of this experiment is to examine the effects in terms of the reduction of the window size of vision span. For this dataset, the character-level tokenization will increase the length of input sequences by 6x on average. In this setting, the fine-tuned Flexible Attention model can achieve a reduction rate of 46% of the vision span. The results indicate that Flexible Attention can automatically adapt to the type of training data and learn to control the strength of penalty properly.

### 5.5 Impact on Real Decoding Speed

In this section, we examine the impact of the reduction of score computation in terms of real decoding speed. We compare the fine-tuned Flexible Attention ($\tau = 1.0$) with the conventional Global Attention on the English-Japanese dataset with character-level tokenization. [6]

We decode 5,000 sentences in the dataset and report the averaged decoding time on both GPU [7] and CPU [8]. For each sequence, the dot production with $\bar{\boldsymbol{h}}_{\boldsymbol{s}}$ in the score function (Eq. 3) is precomputed and cached before decoding. As differ-

[5] In detail, the data points in the plot is based on the thresholds in $(0.3, 0.5, 0.8, 1.0, 1.2, 1.4, 1.6, 5.0, 8.0, 999)$.

[6] For word-level tasks, as the "giant" output layer has large impact on decoding time, we selected the character-level task to measure real decoding speed.

[7] NVIDIA GeForce GTX TITAN X.

[8] Intel Core™ i7-5960X CPU @ 3.00GHz, single core. The implementation uses Theano with openblas as numpy backend.

ent attention models will produce different numbers of output tokens for a same input, that the decoding time will be influenced by different computation steps of the decoder LSTM. In order to fairly compare the decoding time, we force the decoder to use the tokens in the reference as feedbacks. Thus, the number of decoding steps remains the same for both models.

| Model | avg. time (GPU) | avg. time (CPU) |
|---|---|---|
| Global Attention | 123ms | 751ms |
| Flexible Attention | 136ms | 677ms |

Table 3: Average decoding time for one sentence on the English-Japanese dataset with character-level tokenization

As shown in Table 3, reducing the amount of computation in attention model is shown to benefit the decoding speed on CPU. However, applying Flexible Attention slows down the decoding speed on GPU. This is potentially due to the overhead of computing the strength of penalty in Equation 8.

For the CPU-based decoding, after profiling our Theano code, we found that the output layer is the main bottleneck, which accounts for 58% of the computation time. In a recent paper (L'Hostis et al., 2016), the authors show that CPU decoding time can be reduced by 90% by reducing the computation of the output layer, resulting in just over 140ms per sentence. Our proposed attention model has the potential to be combined with their method to further reduce the decoding time on CPU.

As the score function we use in this paper has relatively low computation cost, the difference of real decoding speed is expected to be enlarged with more complicated attention models, such as Recurrent Attention (Yang et al., 2016) and Neural Tensor Network (Socher et al., 2013).

### 5.6 Qualitative Analysis of Flexible Attention

In order to inspect the behaviour of the penalty function in Equation 7, we let the NMT model translate the sentence in Figure 1(a) and record the word positions the attention model considers in each step. The vision span predicted by Flexible Attention is visionized in Figure 1(b).

We can see that the value of $g(t)$ changes dynamically in different context, resulting in different vision span in each step. In the most of the time, the attention is constrained in a local span when translating inside phrases. When emitting the fifth word "TAIRYOU", as the reordering occurs, the attention model looks globally to find the next word to translate. Analyzing the vision spans predicted by Flexible Attention in De-En task also shows similar result that the model only attends to a large span occasionally. The qualitative analysis of Flexible Attention confirms our hypothesis that attending globally in each step is redundant for machine translation. More visualizations can be found in the supplementary material.

## 6 Conclusion

In this paper, we proposed a novel attention framework that is capable of reducing the window size of attention dynamically according to the context. In our experiments, we found the proposed model can safely reduce the window size by 56% for English-Japanese and 64% German-English task on average. For character-based models, our proposed Flexible Attention can also achieve a reduction rate of 46%.

In qualitative analysis, we found that Flexible Attention only needs to put attention on a large window occasionally, especially when long-range reordering is required. The results confirm the existence of massive redundant computation in the conventional attention mechanism. By cutting down unnecessary computation, NMT models can translate extremely long sequence efficiently or incorporate more expensive score functions.

## Acknowledgement

This work is supported by JSPS KAKENHI Grant Number 16H05872.

## References

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

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A character-level decoder without explicit segmentation for neural machine translation. In *ACL*. pages 1693–1703. https://doi.org/10.18653/v1/P16-1160.

Alexandre de Brébisson and Pascal Vincent. 2016. A cheap linear attention mechanism with fast lookups and fixed-size representations. *arXiv preprint arXiv:1609.05866* .

Hideki Isozaki, Tsutomu Hirao, Kevin Duh, Katsuhito Sudoh, and Hajime Tsukada. 2010. Automatic evaluation of translation quality for distant language pairs. In *EMNLP*. pages 944–952.

G Karol, I Danihelka, A Graves, D Rezende, and D Wierstra. 2015. Draw: a recurrent neural network for image generation. In *ICML*.

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

Gurvan L'Hostis, David Grangier, and Michael Auli. 2016. Vocabulary selection strategies for neural machine translation. *arXiv preprint arXiv:1610.00072* .

Liangyou Li, Xiaofeng Wu, Santiago Cortés Vaıllo, Jun Xie, Andy Way, and Qun Liu. 2014. The dcu-ictcas mt system at wmt 2014 on german-english translation task. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*. pages 136–141.

Chin-Yew Lin and Franz Josef Och. 2004. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *ACL*. https://doi.org/10.3115/1218955.1219032.

Thang Luong, Hieu Pham, and D. Christopher Manning. 2015a. Effective approaches to attention-based neural machine translation. In *EMNLP*. pages 1412–1421. https://doi.org/10.18653/v1/D15-1166.

Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. 2015b. Addressing the rare word problem in neural machine translation. In *ACL*. pages 11–19.

Toshiaki Nakazawa, Manabu Yaguchi, Kiyotaka Uchimoto, Masao Utiyama, Eiichiro Sumita, Sadao Kurohashi, and Hitoshi Isahara. 2016. Aspec: Asian scientific paper excerpt corpus. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC 2016)*. pages 2204–2208.

Graham Neubig, Yosuke Nakata, and Shinsuke Mori. 2011. Pointwise prediction for robust, adaptable japanese morphological analysis. In *ACL*. pages 529–533.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *ACL*. pages 1715–1725. https://doi.org/10.18653/v1/P16-1162.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *NIPS*, pages 926–934.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*. pages 3104–3112.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144* .

Zichao Yang, Zhiting Hu, Yuntian Deng, Chris Dyer, and Alex Smola. 2016. Neural machine translation with recurrent attention modeling. *arXiv preprint arXiv:1607.05108* .

## A Supplemental Material

In order to give a better intuition for the conclusion in 5.6, we show a visionization of the vision spans our proposed Flexible Attention predicted on the De-En development corpus. The NMT model can translate well without attending to all positions in each step. In contrast the conventional attention models, Flexible Attention only attends to a large window occasionally.

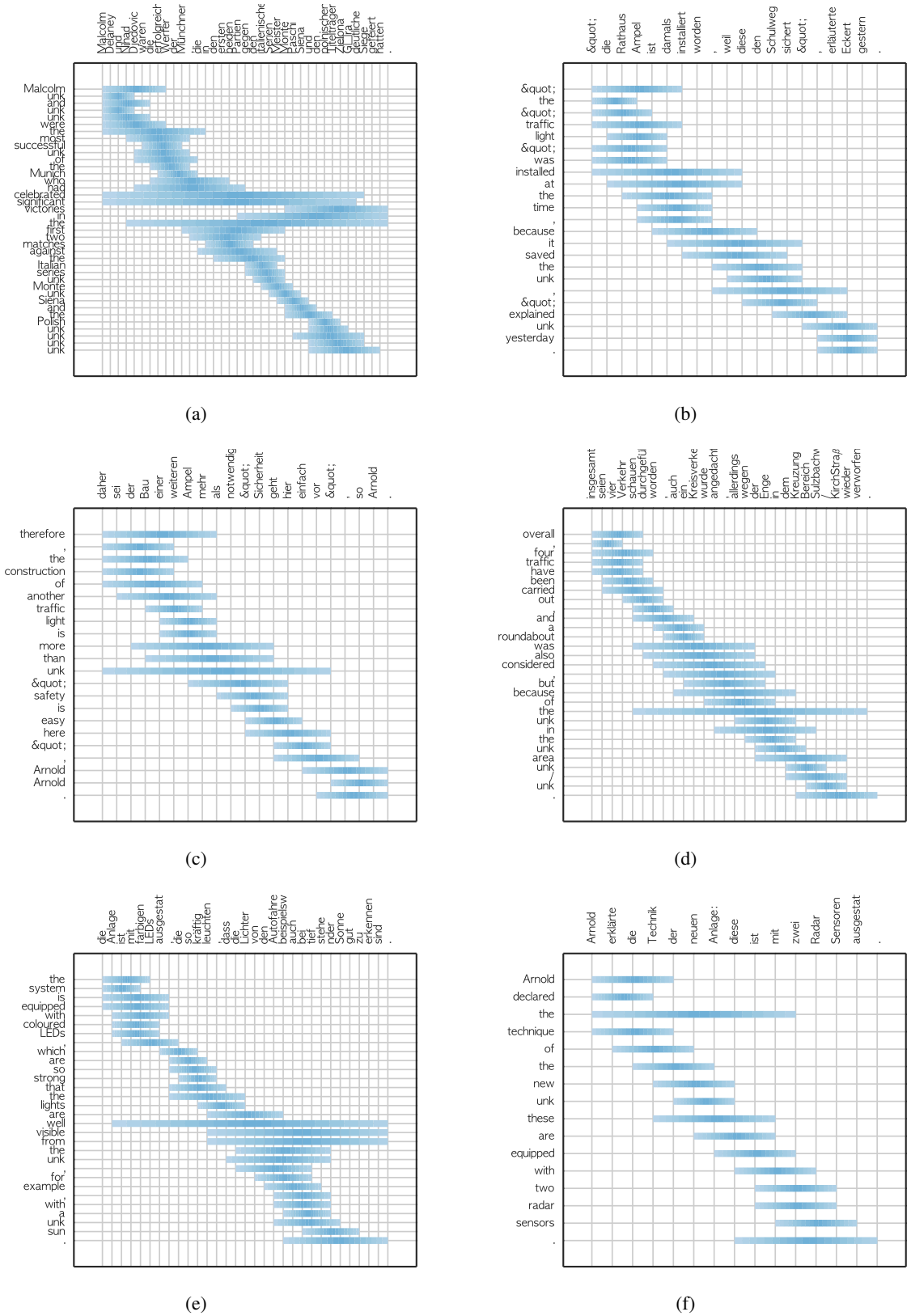


Figure 4: A visionization of the vision spans predicted by Flexible Attention for six random long sentences in the De-En development corpus