Differentiable Window for Dynamic Local Attention

Thanh-Tung Nguyen[*†¶], **Xuan-Phi Nguyen**[*†¶], **Shafiq Joty**[¶§], **Xiaoli Li**[†]
[¶]Nanyang Technological University
[§]Salesforce Research Asia
[†]Institute for Infocomm Research, A-STAR
Singapore
{ng0155ng@e.;nguyenxu002@e.;srjoty@}ntu.edu.sg
xlli@i2r.a-star.edu.sg

Abstract

We propose Differentiable Window, a new neural module and general purpose component for dynamic window selection. While universally applicable, we demonstrate a compelling use case of utilizing Differentiable Window to improve standard attention modules by enabling more focused attentions over the input regions. We propose two variants of Differentiable Window, and integrate them within the Transformer architecture in two novel ways. We evaluate our proposed approach on a myriad of NLP tasks, including machine translation, sentiment analysis, subject-verb agreement and language modeling. Our experimental results demonstrate consistent and sizable improvements across all tasks.

1 Introduction

Computing relative importance across a series of inputs can be regarded as one of the important advances in modern deep learning research. This paradigm, commonly known as *attention* (Bahdanau et al., 2015), has demonstrated immense success across a wide spectrum of applications. To this end, learning to compute contextual representations (Vaswani et al., 2017), to point to the relevant part in the input (Vinyals et al., 2015), or to select windows or spans (Wang and Jiang, 2017) from sequences forms the crux of many modern deep neural architectures.

Despite aggressive advances in developing neural modules for computing relative relevance (Luong et al., 2015; Chiu and Raffel, 2018), there has been no general purpose solution for learning differentiable attention windows. While span selection-based pointer network models typically predict a start boundary and an end boundary (Wang and Jiang, 2017; Seo et al., 2017), these soft predictions generally reside at the last layer of the network and are softly optimized. To the best of our knowledge, there exists no general purpose component for learning differentiable windows within networks.

Although the practical advantages of learning differentiable windows are plenty, this paper focuses on improving attentions with differentiable windows. The key idea is to enable more *focused* attention, leveraging dynamic window selection for limiting (and guiding) the search space for the standard attention modules to work within. This can also be interpreted as performing a form of dynamic local attention.

We make several key technical contributions. First, we formulate the dynamic window selection problem as a problem of learning a *discrete* mask (i.e., binary values representing the window). By learning and composing left and right boundaries, we show that we are able to parameterize the (discrete) masking method. We then propose *soft* adaptations of the above mentioned, namely **trainable soft masking** and **segment-based soft masking**, which are differentiable approximations that can not only be easily optimized in an end-to-end fashion, but also inherit the desirable properties of discrete masking.

While these modules are task and model agnostic, we imbue the state-of-the-art Transformer (Vaswani et al., 2017) model with our differentiable window-based attention. To this end, we propose two further variants, i.e., **multiplicative window attention** and **additive window attention** for improving the Transformer model. Within the context of sequence transduction and self-attention based encoding, learning dynamic attention windows are beneficial because they can potentially eliminate noisy aggregation and alignment from large input sequences. On the other hand, it is good to note that hard attention (Xu et al., 2015b), which replaces the weight average of soft attention with a stochas-

*Equal contributions

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 6589–6599

tic sampling model, tries to achieve similar ends, albeit restricted to token-level selection. Hence, our proposed differentiable windows are more flexible and expressive compared to hard attentions.

We evaluate our Transformer model with differentiable window-based attention on a potpourri of NLP tasks, namely *machine translation, sentiment analysis, language modeling,* and *subject-verb agreement*. Extensive experimental results on these tasks demonstrate the effectiveness of our proposed method. Notably, on the English-German and English-French WMT'14 translation tasks, our method accomplishes improvements of 0.63 and 0.85 BLEU, respectively. On the Stanford Sentiment Treebank and IMDB sentiment analysis tasks, our approach achieves 2.4% and 3.37% improvements in accuracy, respectively. We further report improvements of 0.92% in accuracy and 2.13 points in perplexity on the subject-verb agreement and language modeling tasks, respectively. We make our code publicly available at https://ntunlpsg.github.io/project/dynamic-attention/.

2 Background

The *attention mechanism* enables dynamic selection of relevant contextual representations with respect to a query representation. It has become a key module in most deep learning models for language and image processing tasks, especially in encoder-decoder models (Bahdanau et al., 2015; Luong et al., 2015; Xu et al., 2015a).

2.1 Transformer and Global Attention

The Transformer network (Vaswani et al., 2017) models the encoding and decoding processes using stacked self-attentions and cross-attention (encoder-decoder attentions). Each attention layer uses a scaled multiplicative formulation defined as:

$$\text{score}(\boldsymbol{Q}, \boldsymbol{K}) = \frac{(\boldsymbol{Q}\boldsymbol{W}^Q)(\boldsymbol{K}\boldsymbol{W}^K)^T}{\sqrt{d}} \quad (1)$$

$$\text{att}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \mathcal{S}(\text{score}(\boldsymbol{Q}, \boldsymbol{K}))(\boldsymbol{V}\boldsymbol{W}^V) \quad (2)$$

where $\mathcal{S}(\boldsymbol{A})$ denotes the *softmax* operation over each row of matrix $\boldsymbol{A}$, $\boldsymbol{Q} \in \mathbb{R}^{n_q \times d}$ is the matrix containing the n_q query vectors, and $\boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{n \times d}$ are the matrices containing the n key and value vectors respectively, with d being the number of vector dimensions; $\boldsymbol{W}^Q$, $\boldsymbol{W}^K$, $\boldsymbol{W}^V \in \mathbb{R}^{d \times d}$ are the associated weights to perform linear transformations.

To encode a source sequence, the encoder applies *self-attention*, where $\boldsymbol{Q}$, $\boldsymbol{K}$ and $\boldsymbol{V}$ contain the same vectors coming from the output of the previous layer.[1] In the decoder, each layer first applies *masked self-attention* over previous-layer states. The resulting vectors are then used as queries to compute *cross-attentions* over the encoder states. For cross-attention, $\boldsymbol{Q}$ comprises the decoder self-attention states while $\boldsymbol{K}$ and $\boldsymbol{V}$ contain the encoder states. The attention mechanism adopted in the Transformer is considered **global** since the attention context spans the entire sequence.

2.2 Windows in Attentions

In theory, given enough training data, global attention should be able to model dependencies between the query and the key vectors well. However, in practice we have access to only a limited amount of training data. Several recent studies suggest that incorporating more focused attention over important local regions in the input sequence as an explicit inductive bias could be more beneficial.

In particular, Shaw et al. (2018) show that adding relative positional biases to the attention scores (Eq. 1) increases BLEU scores in machine translation. Specifically, for each query $\boldsymbol{q}_i \in \boldsymbol{Q}$ at position i and key $\boldsymbol{k}_j \in \boldsymbol{K}$ at position j, a trainable vector $\boldsymbol{a}_{i,j} = \boldsymbol{w}_{max(-\tau, min(j-i, \tau))}$ is added to the key vector before the query-key dot product is performed. The window size τ is chosen via tuning. Sperber et al. (2018) also consider local information by restricting self-attention to neighboring representations to improve long-sequence acoustic modeling. Although shown to be effective, their methods only apply to self-attention and not to cross-attention where the query vectors come from a different sequence.

That said, Luong et al. (2015) are the first to propose a Gaussian-based local attention for *cross-attention*. At each decoding step t, their model approximates the source-side pivot position p_t as a function of the decoding state and the source sequence length. Then, local attention is achieved by multiplying the attention score with a *confidence* term derived from a $\mathcal{N}(p_t, \sigma^2)$ distribution. The aligned pivot p_t and the variance σ^2 (a hyper-parameter) respectively represent the center and the size of the local window.

[1] Initially, $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ contain the token embeddings.

Meanwhile, Yang et al. (2018) improve the method of Luong et al. (2015) by assigning a soft window weight (a Gaussian bias) to obtain a flexible window span. Despite effective, the aligned pivot position in the source is determined only by the decoder state, while the encoder states are disregarded - these should arguably give more relevant information regarding the attention spans over the source sequence. Besides, the confidence for local attention span may not strictly follow a normal distribution, but rather vary dynamically depending on the relationship between the query and the key. Furthermore, the approach of Luong et al. (2015) is only applicable to cross-attention while the one of Yang et al. (2018) works better only for *encoder* self-attention as shown in their experiments.

Our proposed *differentiable window* approach to local attention addresses the above limitations of previous methods. Specifically, our methods are dynamic and applicable to encoder and decoder self-attentions as well as cross-attention, without any functional constraints. They incorporate encoder states into the local window derivation. They are also invariant to sequence length, which removes the dependence on *global* features from the local context extraction process.

3 Dynamic Differentiable Window

Our proposed attention method works in two steps: (*i*) derive the attention span for each query vector to attend over, and (*ii*) compute the respective attention vector using the span. In this section, we present our approaches to step (*i*) by proposing *trainable soft masking* and *segment-based soft masking*. In the next section, we present our methods to compute the attention vectors. To give the necessary background to understand what can be expected from our method, we first present the *discrete masking* case.

3.1 Discrete Window Masking

In this context, we seek to dynamically derive a *boolean mask* vector for each query that will indicate the window in the key-sequence over which the query should attend. In other words, attentions are only activated on the consecutive positions where the mask vector element is 1, and the positions with 0 are canceled out. Let the query vector and the key-sequence be $\boldsymbol{q} \in \mathbb{R}^d$ and $\boldsymbol{K} = (\boldsymbol{k}_1, \boldsymbol{k}_2, \ldots, \boldsymbol{k}_n)$, respectively. Formally, we define the local attention mask vector $\boldsymbol{m}_q \in \{0,1\}^n$ for the query $\boldsymbol{q}$ as

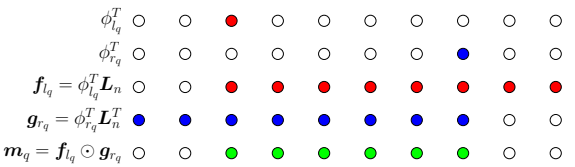


Figure 1: Example of ϕ, $\boldsymbol{f}$, and $\boldsymbol{g}$ vectors and how the mask vector $\boldsymbol{m}_q$ can be derived for $l_q = 3$ and $r_q = 8$.

follows.

$$\boldsymbol{m}_q^i = \begin{cases} 1, & \text{if } l_q \leq i \leq r_q \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

where l_q and r_q denote the left and right positional indices that form a *discrete* window $[l_q, r_q]$ over which the query attends. As such, in the standard *global* attention, $l_q = 1$ and $r_q = n$ for all the query vectors, and in decoder self-attention, $l_q = 1$ and $r_q = t$ for the query vector at decoding step t. To facilitate the construction of $\boldsymbol{m}_q$, we first define vectors ϕ_k, $\boldsymbol{f}_k$, $\boldsymbol{g}_k$ and matrix $\boldsymbol{L}_n$ with entries as:

$$\phi_k^i = \begin{cases} 1, & \text{if } i = k \\ 0, & \text{otherwise} \end{cases}; \quad \boldsymbol{f}_k^i = \begin{cases} 1, & \text{if } i \geq k \\ 0, & \text{otherwise} \end{cases}$$

$$\boldsymbol{g}_k^i = \begin{cases} 1, & \text{if } i \leq k \\ 0, & \text{otherwise} \end{cases}; \quad \boldsymbol{L}_n^{i,j} = \begin{cases} 1, & \text{if } i \leq j \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

where $\phi_k \in \{0,1\}^n$ denotes the one-hot representation for a boundary position k (from the left or right of a sequence), and $\boldsymbol{f}_k, \boldsymbol{g}_k \in \{0,1\}^n$ are the 'rightward' mask vector and 'leftward' mask vector, respectively; $\boldsymbol{L}_n \in \{0,1\}^{n \times n}$ denotes a unit-value (1) upper-triangular matrix with i and j being the row and column indices respectively. Figure 1 visualizes how these entities appear. Specifically, $\boldsymbol{f}_k$ has entry values of 1's for position k and its right positions, while $\boldsymbol{g}_k$ has entry values of 1's for position k and its left positions. As such, $\boldsymbol{f}_k$ and $\boldsymbol{g}_k$ can be derived from ϕ_k and $\boldsymbol{L}_n$ as follows.

$$\boldsymbol{f}_k = \phi_k^T \boldsymbol{L}_n; \quad \boldsymbol{g}_k = \phi_k^T \boldsymbol{L}_n^T \tag{5}$$

Note that $\boldsymbol{f}_k$ can be interpreted as the *cumulative sum* across ϕ_k, while $\boldsymbol{g}_k$ as the *inverse cumulative sum* across ϕ_k.

Given the above definitions, the mask vector $\boldsymbol{m}_q$ for a query $\boldsymbol{q}$ to attend over the window $[l_q, r_q]$ in

the key sequence such that $1 \leq l_q \leq r_q \leq n$ can be achieved by:

$$\boldsymbol{m}_q = \boldsymbol{f}_{l_q} \odot \boldsymbol{g}_{r_q} = (\phi_{l_q}^T \boldsymbol{L}_n) \odot (\phi_{r_q}^T \boldsymbol{L}_n^T) \quad (6)$$

where $\odot$ denotes element-wise multiplication. As shown in Figure 1, $\boldsymbol{m}_q$ represents the intersection between $\boldsymbol{f}_{l_q}$ and $\boldsymbol{g}_{r_q}$, and forms a masking span for the attention.

3.2 Trainable Soft Masking

The above masking method is non-differentiable as ϕ is discrete, which makes it unsuitable in an end-to-end neural architecture. In our trainable soft masking method, we approximate the discrete one-hot vector ϕ with a pointing mechanism (Vinyals et al., 2015).[2] Specifically, given the query $\boldsymbol{q}$ and the key-sequence $\boldsymbol{K}$ as before, we define confidence vectors $\hat{\phi}_{l_q}, \hat{\phi}_{r_q} \in \mathbb{R}^n$ as follows.

$$\hat{\phi}_{l_q} = \mathcal{S}(\frac{\boldsymbol{q}^T \boldsymbol{W}_L^Q (\boldsymbol{K} \boldsymbol{W}_L^K)^T}{\sqrt{d}}) \quad (7)$$

$$\hat{\phi}_{r_q} = \mathcal{S}(\frac{\boldsymbol{q}^T \boldsymbol{W}_R^Q (\boldsymbol{K} \boldsymbol{W}_R^K)^T}{\sqrt{d}}) \quad (8)$$

where $\mathcal{S}$ is the *softmax* function as defined before, and $\boldsymbol{W}_L^Q, \boldsymbol{W}_L^K, \boldsymbol{W}_R^Q, \boldsymbol{W}_R^K \in \mathbb{R}^{d \times d}$ are trainable parameters. Eq. 7-8 approximate the left and right boundary positions of the mask vector for the query $\boldsymbol{q}$. However, contrary to the discrete case, they do not enforce *absolute* cancellation or activation of attention weights on any position in the key-sequence. Instead, they assign a confidence score to each position. This allows the model to gradually correct itself from invalid assignments. Moreover, the *softmax* operations enable differentiability while maintaining the gradient flow in an end-to-end neural architecture.

Note however that the left and right boundary concepts have now become ambiguous since the positions $l_q = \arg\max(\hat{\phi}_{l_q})$ and $r_q = \arg\max(\hat{\phi}_{r_q})$ are not guaranteed to conform to the constraint $l_q \leq r_q$. To understand its implication, lets first consider the discrete case in Eq. 6; the element-wise multiplication between $\boldsymbol{f}_{l_q}$ and $\boldsymbol{g}_{r_q}$ results in a *zero* vector for $\boldsymbol{m}_q$ if $l_q > r_q$, canceling out the attention scores entirely. Although not absolute *zeros*, in the continuous case, $\boldsymbol{m}_q$ would potentially contain significantly small values, which renders the attention implausible. To address this, we compute the *soft* mask vector $\hat{\boldsymbol{m}}_q$ as follows.

$$\hat{\boldsymbol{m}}_q = (\hat{\phi}_{l_q}^T \boldsymbol{L}_n) \odot (\hat{\phi}_{r_q}^T \boldsymbol{L}_n^T) + (\hat{\phi}_{r_q}^T \boldsymbol{L}_n) \odot (\hat{\phi}_{l_q}^T \boldsymbol{L}_n^T) \quad (9)$$

This formulation has two additive terms; the former constructs the mask vector when $l_q \leq r_q$, whereas the latter is activated when $l_q > r_q$. This ensures a non-zero result regardless of l_q and r_q values. It can be shown that the values in $\hat{\boldsymbol{m}}_q$ represent the expected value of the discrete flags in $\boldsymbol{m}_q$, i.e., $\hat{\boldsymbol{m}}_q = \mathbb{E}(\boldsymbol{m}_q)$; see Appendix for a proof.

We concatenate the mask vectors horizontally for all the query vectors in $\boldsymbol{Q} \in \mathbb{R}^{m \times d}$ to get the mask matrix $\boldsymbol{M} \in \mathbb{R}^{m \times n}$. Since the pointing mechanism is invariant to sequence length, the computation of the mask vectors enjoys the same advantages, enabling our models to efficiently perform attentions on any arbitrarily long sequences. In addition, the method is applicable to all attention scenarios – from decoder to encoder cross-attention, encoder self-attention, and decoder self-attention.

3.3 Segment-Based Soft Masking

The soft masking introduced above modulates the attention weight on each token separately which may result in unsmooth attention weights on neighbouring tokens. However, words in a sentence are related and they often appear in chunks or phrases, contributing to a shared meaning. Thus, it may be beneficial to assign identical mask values to the tokens within a segment so that they are equally treated in the window selection method. In this section, we propose a novel extension to our soft masking method that enables the mask vector to share the same masking values for the tokens within a segment in a key-sequence.

The main idea is to divide the key-sequence $\boldsymbol{K} = (\boldsymbol{k}_1, \boldsymbol{k}_2, \ldots, \boldsymbol{k}_n)$ into $\lceil n/b \rceil$ consecutive segments and to assign the same masking value to the tokens in a segment. The segment size b is considered a hyper-parameter. We compute the segment-based mask vector $\boldsymbol{m}'_q$ similarly as in Eq. 9, but with $\boldsymbol{L}_n$ replaced by $\boldsymbol{J}_n \in \mathbb{R}^{n \times n}$ defined as follows.

$$\boldsymbol{J}_n^{i,j} = \begin{cases} 1, & \text{if } i \leq b\lceil \frac{j}{b} \rceil \\ 0, & \text{otherwise} \end{cases} \quad (10)$$

[2] However, unlike the standard pointer network, in our case there is no direct supervision for learning the pointing function. Our network instead learns it from the end prediction task.

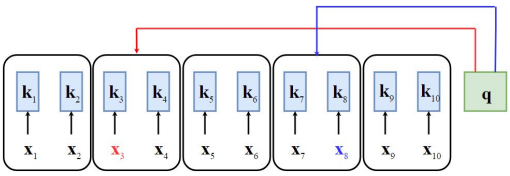


Figure 2: Segment-based masking for segment size = 2. Instead of pointing to the left and right indices of the *tokens*, the soft segment-based method (approximately) points to the left and right boundaries of the *segments*, respectively.

$$\boldsymbol{m}'_q = (\hat{\phi}_{l_q}^T \boldsymbol{J}_n) \odot (\hat{\phi}_{r_q}^T \boldsymbol{J}_n^T) + (\hat{\phi}_{r_q}^T \boldsymbol{J}_n) \odot (\hat{\phi}_{l_q}^T \boldsymbol{J}_n^T) \tag{11}$$

Eq. 10 - 11 ensure that all the items in a segment share the same masking value, which is the cumulative sum of the confidence scores in $\hat{\phi}_{l_q}$ and $\hat{\phi}_{r_q}$. For instance, suppose $\hat{\phi}_{l_q} = (a_1, a_2, a_3, \ldots, a_n)$ and segment size $b = 2$, then the term $\hat{\phi}_{l_q}^T \boldsymbol{J}_n$ evaluates to $(\sum_{i=1}^{2} a_i, \sum_{i=1}^{2} a_i, \sum_{i=1}^{4} a_i, \ldots)$, and $\hat{\phi}_{l_q}^T \boldsymbol{J}_n^T$ evaluates to $(\sum_{i=1}^{n} a_i, \sum_{i=1}^{n} a_i, \sum_{i=3}^{n} a_i, \ldots)$. Similarly, $\hat{\phi}_{r_q}^T \boldsymbol{J}_n^T$ and $\hat{\phi}_{r_q}^T \boldsymbol{J}_n$ will have segment-level effects on the cumulative sums. Figure 2 visualizes the method with an example for $b = 2$.

One advantage of this approach is that it allows us to control the masking behavior (by varying b) without increasing the number of parameters compared to the token-based masking. We also show its effectiveness in our experiments.

4 Dynamic Window Attention Methods

Having presented our method to compute the mask vector that defines the attention spans, we now present our methods to incorporate the mask vectors into the attention layers.

4.1 Multiplicative Window Attention

In this approach, the attention weights (Eq. 2) are (element-wise) multiplied by the mask matrix $\boldsymbol{M}$ to confine their attention scope defined by the mask. Formally, the attention scores and outputs are defined as follows.

$$\text{score} = \frac{(\boldsymbol{Q}\boldsymbol{W}^Q)(\boldsymbol{K}\boldsymbol{W}^K)^T}{\sqrt{d}} \tag{12}$$

$$\text{att}_{\text{MW}} = (\mathcal{S}(\text{score}) \odot \boldsymbol{M})(\boldsymbol{V}\boldsymbol{W}^V) \tag{13}$$

In this approach, the standard global attention weights are suppressed and partially overshadowed by the attention window imposed by $\boldsymbol{M}$. Thus, it can be interpreted as a *local attention* method similar to Luong et al. (2015). However, instead of using a static Gaussian bias, we use a dynamic mask to modulate the attention weights.

4.2 Additive Window Attention

Having a *local* attention window could be beneficial, but it does not rule out the necessity of global attention, which has been shown effective in many applications (Vaswani et al., 2017; Devlin et al., 2019). Thus, we also propose an *additive window attention*, which implements a combination of global attention and local attention. The attention output in this method is formally defined as

$$s_{\text{glb}} = (\boldsymbol{Q}\boldsymbol{W}_{\text{glb}}^Q)(\boldsymbol{Q}\boldsymbol{W}_{\text{glb}}^K)^T \tag{14}$$

$$s_{\text{loc}} = (\boldsymbol{Q}\boldsymbol{W}_{\text{loc}}^Q)(\boldsymbol{Q}\boldsymbol{W}_{\text{loc}}^K)^T \odot \boldsymbol{M} \tag{15}$$

$$\text{score}_{\text{AW}} = \frac{s_{glb} + s_{loc}}{\sqrt{d}} \tag{16}$$

$$\text{att}_{\text{AW}} = \mathcal{S}(\text{score}_{\text{AW}})(\boldsymbol{V}\boldsymbol{W}^V) \tag{17}$$

where $\boldsymbol{W}_{\text{glb}}^Q, \boldsymbol{W}_{\text{glb}}^K, \boldsymbol{W}_{\text{loc}}^Q$, and $\boldsymbol{W}_{\text{loc}}^K \in \mathbb{R}^{d \times d}$ are the weight matrices for global and local attentions.

Compared to the multiplicative window attention where the mask re-evaluates the global attention weights, additive window attention applies the mask vector to the *local attention scores* (s_{loc}), which is then added to the *global attention scores* (s_{glb}) before passing it through the softmax function. In this way, the mask-defined local window does not suppress the global context but rather complements it with a local context. Moreover, the resulting attention weights add up to *one*, which avoids attention weights diminishment that could occur in the multiplicative window attention. Additive merger of global and local window components may also facilitate more stable gradient flows.

4.3 Implementation in Transformer

We now describe how the proposed dynamic window attention methods can be integrated into the Transformer.

Encoder, Decoder and Cross Attentions. Our proposed methods can be readily applied to the any of the attention layers in the Transformer framework. We could also selectively apply our methods to different layers in the encoder and decoder.

In our initial experiments on WMT'14 English-German development set, we observed that the following settings provide more promising performance gains.

First, encoder self-attention layers benefit most from *additive window attention*, while decoder self-attention layers prefer *multiplicative attention*. This shows that the *global* attention component is more useful when the key sequence is provided entirely in the encoder, while less useful when only the fragmented key sequence (past keys) is visible in the decoder. Second, the above argument is further reinforced as we found that cross-attention layers also prefer *additive window attention*, where the entire source sequence is available. Third, cross-attention works better with *segment-based masking*, which provides smoothness and facilitates phrase (n-gram) based translations.

Lower-layer Local Attentions. It has been shown that deep neural models learn simple word features and local syntax in the lower layers, while higher layers learn more complex context-dependent aspects of word semantics. Belinkov et al. (2017) show this on NMT models, while Peters et al. (2018) and Jawahar et al. (2019) show this on representation learning with ELMo and BERT respectively. In other words, local contextual information can still be derived in higher layers with the standard global attention. As such, we propose to apply our dynamic window attention methods only to the first 3 layers of the Transformer network, leaving the top 3 layers intact. Our diverse experiments in the following section support this setup as it offers substantial improvements, whereas using local attention in higher layers does not show gains, but rather increases model parameters.

5 Experiment

In this section, we present the training settings, experimental results and analysis of our models in comparison with the baselines on machine translation (MT), sentiment analysis, subject verb agreement and language modeling (LM) tasks.

5.1 Machine Translation

We trained our models on the standard WMT'16 English-German (En-De) and WMT'14 English-French (En-Fr) datasets containing about 4.5 and 36 million sentence pairs, respectively. For validation (development) purposes, we used *newstest2013* for En-De and a random split from the training set for En-Fr. All translation tasks were evaluated against their respective *newstest2014* test sets, in case-sensitive tokenized BLEU. We used *byte-pair encoding* (Sennrich et al., 2016) with shared source-target vocabularies of 32,768 and 40,000 sub-words for En-De and En-Fr translation tasks, respectively. We compare our models with three strong baselines: (*i*) Transformer Base (Vaswani et al., 2017), (*ii*) Transformer Base with Relative Position (Shaw et al., 2018), and (*ii*) Transformer Base with Localness Modeling (Yang et al., 2018). To ensure a fair comparison, we trained our models and the baselines with the following training setup.

Training Setup. We followed model specifications in (Vaswani et al., 2017) and optimization settings in (Ott et al., 2018), with some minor modifications. Specifically, we used word embeddings of dimension 512, feedforward layers with inner dimension 2048, and multi-headed attentions with 8 heads. We trained our models on a single physical GPU but replicated the 8-GPU setup following the *gradient aggregation* method proposed by Ott et al. (2018). We trained the models for 200,000 *updates* for En-De and 150,000 *updates* for En-Fr translation tasks. Finally, we averaged the last 5 checkpoints to obtain the final models for evaluation. The segment size b in the segment-based masking method was set to 5.[3]

Translation Results. We report our translation results in Table 1; **Enc(AW)** indicates the use of *additive window* (AW) attention in the encoder, **Dec(MW)** indicates the use of *multiplicative window* (MW) attention in the decoder, and **Cr(AW,Seg)** indicates the use of additive window attention with *segment-based masking* for cross-attention. The attention module that is not specified in our naming convention uses the default token-based global attention in the Transformer. For example, **Enc(AW)-Dec(MW)** refers to the model that uses AW attention in the encoder, MW attention in the decoder and the default global attention for cross attention.

We notice that despite a minor increase in the number of parameters, applying our attentions in the encoder and decoder offers about 0.7 and 1.0 BLEU improvements in En-De and En-Fr translation tasks respectively, compared to the

[3]We did not tune b; tuning b might improve the results further.

Model	#-params	En-De	En-Fr
Vaswani et al. (2017)	63M	27.46	39.21
Shaw et al. (2018)	63M	27.56	39.37
Yang et al. (2018)	63M	27.62	39.47
Our Models			
Enc(AW)-Dec(MW)	68M	28.11	40.24
Cr(AW, Seg)	65M	28.13	40.06
Enc(AW)-Cr(AW,Seg)-Dec(MW)	73M	**28.25**	**40.32**

Table 1: BLEU scores for different models in WMT'14 English-German and English-French translation tasks.

Method	Module	Full (6 layers)	Partial (3 layers)
Transformer	-	27.46	-
AW	Encoder	27.77	27.90
MW	Encoder	27.25	27.40
AW	Decoder	27.73	27.85
MW	Decoder	27.88	28.04
AW	Cross	27.78	27.97
MW	Cross	27.58	27.79

Table 2: Evaluation of Additive Window (AW) and Multiplicative Window (MW) attentions in encoder/decoder self attention and cross attention for full vs. partial settings.

Transformer base (Vaswani et al., 2017). Our model with the segment-based additive method for *cross* attention achieves a similar performance. We observe further improvements as we apply our attentions in all the attention modules of the Transformer. Specifically, our model Enc(AW)-Cr(AW,Seg)-Dec(MW) achieves 28.25 and 40.32 BLEU in En-De and En-Fr translation tasks, outperforming Transformer base with localness (Yang et al., 2018) by 0.63 and 0.85 BLEU, respectively.

5.2 Ablation Study

To verify our modeling decisions, we performed an ablation study in the WMT'14 En-De translation task. In particular, we evaluated (*i*) the impact of applying our differentiable window attentions in all layers vs. only in certain lower layers of the Transformer network, (*ii*) which window attention methods (additive or multiplicative) are suitable particularly for the encoder/decoder self-attention and cross-attention, and (*iii*) the impact of segment-based masking in different attention modules. (*iv*) training efficiency and performance of our best model with the similar models. Plus, to further interpret our window-based attention, we also provide the local window visualization.

Full vs. Partial. Table 2 shows BLEU scores for the Transformer models that employ our window-based attentions in all 6 layers (**Full**) vs. only in the first 3 layers (**Partial**), as well as the methods used in different attention modules (encoder/decoder self-attention, cross-attention). We can see that almost all the models with window-based methods in the first 3 layers outperform those that use them in all 6 layers. This gives the setup significant advantages as it performs not only better in BLEU but also requires less parameters.

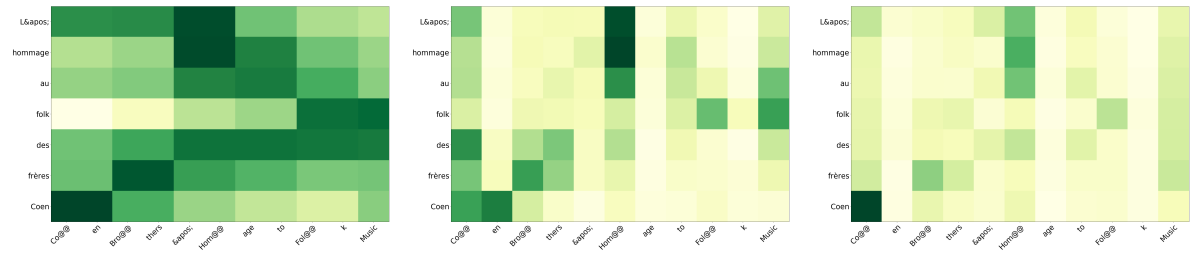
The results also show that multiplicative window (MW) attention is preferred in decoder self-attention, while additive window (AW) is more suitable for encoder self-attention and for cross-attention. This suggests that the global context, which is maintained in AW, is more useful when it is entirely available like in encoder self-attention and cross attention. In contrast, incomplete and partially-generated context in decoder self-attention may induce more noise than information, where MW attention renders better performance than AW.

Model	Token-based	Segment-based
Cr(AW)	27.97	28.13
Enc(AW)-Dec(MW)	28.11	27.91

Table 3: BLEU scores for token- and segment-based masking in cross attention and encoder self-attention. The decoder self-attention always uses token-based masking.

Token- vs. Segment-based. Table 3 compares the results for using token-based vs. segment-based masking methods in different attention modules of the network. Note that it is preferred for decoder self-attention to adopt token-based masking since the decoder cannot point to *unfinished* segments in autoregressive generation, if it had used segment-based masking. We see that segment-based additive window masking outdoes its token-based counterpart (28.13 vs. 27.97 BLEU) for cross-attention. Meanwhile, for encoder self-attention, token-based masking performs better than segment-based masking by 0.2 BLEU. This suggests that segments (or phrases) represent better *translation* units than tokens, justifying its performance superiority in cross-lingual attention but not in monolingual (self-attention) encoding.

Speed and Parameters. As shown in table 4, our training efficiency is competitive to the baselines. That is, the training speed for our model is 1.04



(a) Local masking scores ($\boldsymbol{M}$). (b) Our attention scores. (c) Transformer attention scores.

Figure 3: Visualization of masking scores, and attention scores for our and the original Transformer models.

Model	#-params	# steps/sec	BLEU
Vaswani et al. (2017)	63M	1.20	27.46
Yang et al. (2018)	63M	1.07	27.62
Vaswani et al. (2017) 7 layers	69M	1.05	27.74
Vaswani et al. (2017) 8 layers	75M	0.99	27.89
Enc(AW)-Cr(AW,Seg)-Dec(MW)	73M	1.04	**28.25**

Table 4: Training efficiency and size of similar models

steps/sec which is similar to Yang et al. (2018). Besides, our model outperforms the Transformer with 8 layers, which has more parameters. This suggests that our performance gain may not come from additional parameters, but rather from a better inductive bias through the dynamic window attention.

Local Window Visualization. To further interpret our window-based attentions, Figure 3a shows the cross-attention soft masking values ($\hat{\boldsymbol{m}}_q$) on the source tokens for each target token in an En-Fr test sample assigned by our Enc(AW)-Cr(AW,Seg)-Dec(MW) model. The darker the score, the higher the attention is from a target token to a source token. We can see the relevant subwords are captured by the attentions quite well, which promotes ngram-level alignments. For instance, the mask ($\hat{\boldsymbol{m}}_q$) guides the model to evenly distribute attention scores on sub-words "Co@@" and "en" (Fig. 3b), while standard attention is biased towards "Co@@" (Fig. 3c). Similar phenomenon can be seen for "Bro@@" and "thers" (towards "frères").

5.3 Text Classification

We evaluate our models on the Stanford Sentiment Treebank (SST) (Socher et al., 2013), IMDB sentiment analysis (Maas et al., 2011) and Subject-Verb Aggreement (SVA) (Linzen et al., 2016) tasks. We compare our attention methods (incorporated into the Transformer encoder) with the encoders of Vaswani et al. (2017), Shaw et al. (2018) and Yang et al. (2018).

Model	STT	IMDB	SVA
Vaswani et al. (2017)	79.36	83.65	94.48
Shaw et al. (2018)	79.73	84.61	95.27
Yang et al. (2018)	79.24	84.13	95.00
Enc (MW)	79.70	85.09	95.95
Enc (AW)	**82.13**	**87.98**	**96.19**

Table 5: Classification accuracy on on Stanford Sentiment Treebank (SST) and IMDB sentiment analysis and Subject-Verb Agreement(SVA) tasks.

Training Setup. As the datasets are quite small compared to the MT datasets, we used *tiny versions* of our models as well as the baselines.[4] Specifically, the models consist of a 2-layer Transformer encoder with 4 attention heads, 128 hidden dimensions and 512 feedforward inner dimensions. In these experiments, our attention methods are applied only to the first layer of the network. We trained for 3,000, 10,000 and 10,000 updates for SST, IMDB and SVA tasks, respectively on a single GPU machine.

Results. Table 5 shows the results. Our multiplicative window approach (Enc (MW)) achieves up to 79.7%, 85.1% and 95.95% accuracy in SST, IMDB and SVA, exceeding Transformer (Vaswani et al., 2017) by 0.4%, 1.35% and 1.47%, respectively. Our additive window attention (Enc (AW)) renders even more improvements. Specifically, it outperforms Transformer with relative position (Shaw et al. 2018) by 2.4% and 3.37%, 0.92% reaching 82.13%, 87.98% and 96.19% accuracy in SST, IMDB and SVA, respectively. In fact, the results demonstrate consistent trends with our earlier MT experiments: additive window attention outdoes its multiplicative counterpart in the encoder,

[4] As specified in https://github.com/tensorflow/tensor2tensor.

Model	Perplexity
Vaswani et al. (2017)	46.37
Shaw et al. (2018)	46.13
Dec (MW)	**44.00**
Dec (AW)	44.95

Table 6: Perplexity scores on 1-billion-word language modeling benchmark (the lower the better).

where the entire key sequence is available.

5.4 Language Modeling

Finally, to demonstrate our proposed methods as effective general purpose NLP components, we evaluate them on the One Billion Word LM Benchmark dataset (Chelba et al., 2013). The dataset contains 768 million words of data compiled from WMT 2011 News Crawl data, with a vocabulary of 32,000 words. We used its held-out data as the test set.

Training Setup. As the LM dataset is considerably large, we used the same model settings as adopted in our MT experiments. For these experiments, we only trained the models on virtually 4 GPUs for 100,000 *updates* using gradient aggregation on a single GPU machine. Note that only the self-attention based autoregressive decoder of the Transformer framework is used in this task. Therefore, the method of Yang et al. (2018) is not applicable to this task.

Results. Table 6 shows the perplexity scores. As can be seen, our multiplicative and additive window attention models both surpass Transformer (Vaswani et al., 2017) by 2.37 and 1.42 points respectively, reaching 44.00 and 44.95 perplexity scores respectively. In addition, it is noteworthy that similar to MT experiments, multiplicative attention outperforms the additive one on this task, where the decoder is used. This further reinforces the claim that where the global context is not *fully* available like in the decoder, the incomplete global context may induce noises into the model. Thus, it is effective to embrace dynamic local window attention to suppress the global context, for which the multiplicative window attention is designed.

6 Conclusion

We have presented a novel Differential Window method for dynamic window selection, and used it to improve the standard attention modules by enabling more focused attentions. Specifically, we proposed Trainable Soft Masking and Segment-based Masking, which can be applied to encoder/decoder self-attentions and cross attention.

We evaluated our models on four NLP tasks including machine translation, sentiment analysis, subject verb agreement and language modeling. Our experiments show that our proposed methods outperform the baselines significantly across all the tasks. All in all, we demonstrate the benefit of incorporating the differentiable window in the attention. In the future, we would like to extend our work to make a syntactically-aware window that can automatically learn tree (or phrase) structures.

Acknowledgments

We would like to express our gratitude to Yi Tay and our anonymous reviewers for their insightful feedback on our paper. Shafiq Joty would like to thank the funding support from his Start-up Grant (M4082038.020).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017. What do neural machine translation models learn about morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 861–872, Vancouver, Canada. Association for Computational Linguistics.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. Technical report, Google.

Chung-Cheng Chiu and Colin Raffel. 2018. Monotonic chunkwise attention. In *International Conference on Learning Representations*.

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

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure

of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy. Association for Computational Linguistics.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of lstms to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, EMNLP, pages 1412–1421. ACL.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. 2018. Scaling neural machine translation. In *Proceedings of the Third Conference on Machine Translation (WMT)*.

Matthew Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. 2018. Dissecting contextual word embeddings: Architecture and representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509, Brussels, Belgium. Association for Computational Linguistics.

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

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642. Association for Computational Linguistics.

Matthias Sperber, Jan Niehues, Graham Neubig, Sebastian Stuker, and Alex Waibel. 2018. Self-attentional acoustic models. In *Interspeech 2018, 19th Annual Conference of the International Speech Communication Association, Hyderabad, India, 2-6 September 2018.*, pages 3723–3727.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2692–2700. Curran Associates, Inc.

Shuohang Wang and Jing Jiang. 2017. Machine comprehension using match-lstm and answer pointer. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015a. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2048–2057, Lille, France. PMLR.

Kelvin Xu, Jimmy Lei Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015b. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 2048–2057. JMLR.org.

Baosong Yang, Zhaopeng Tu, Derek F. Wong, Fandong Meng, Lidia S. Chao, and Tong Zhang. 2018. Modeling localness for self-attention networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4449–4458, Brussels, Belgium. Association for Computational Linguistics.

Appendix

Proof: $\hat{\boldsymbol{m}}_q = \mathbb{E}(\boldsymbol{m}_q)$

The probability of left and right boundary for a query q:

$$\hat{\phi}_{l_q} = \mathcal{S}(\frac{\boldsymbol{q}^T \boldsymbol{W}_L^Q (\boldsymbol{K}\boldsymbol{W}_L^K)^T}{\sqrt{d}}) \quad (18)$$

$$\hat{\phi}_{r_q} = \mathcal{S}(\frac{\boldsymbol{q}^T \boldsymbol{W}_R^Q (\boldsymbol{K}\boldsymbol{W}_R^K)^T}{\sqrt{d}}) \quad (19)$$

For any k,

$$p(f_k = 1) = p(l_q \leq k) = \sum_{\hat{\phi}_{l_q} \leq k} \hat{\phi}_{l_q} = (\hat{\phi}_{l_q}^T \boldsymbol{L}_n)_k \quad (20)$$

$$p(g_k = 1) = p(r_q \geq k) = \sum_{\hat{\phi}_{r_q} \geq k} \hat{\phi}_{r_q} = (\hat{\phi}_{r_q}^T \boldsymbol{L}_n^T)_k \quad (21)$$

Since f_k and g_k are binary values,

$$\hat{f}_k = p(f_k = 1) = \mathbb{E}(f_k) \quad (22)$$

$$\hat{g}_k = p(g_k = 1) = \mathbb{E}(g_k) \quad (23)$$

Hence,

$$\hat{\boldsymbol{m}}_q = \hat{\boldsymbol{f}}_{l_q} \odot \hat{\boldsymbol{g}}_{r_q} + \hat{\boldsymbol{f}}_{r_q} \odot \hat{\boldsymbol{g}}_{l_q} = \mathbb{E}(\boldsymbol{m}_q) \quad (24)$$