
Time-aware Large Kernel Convolutions

Vasileios Lioutas¹ Yuhong Guo¹

Abstract

To date, most state-of-the-art sequence modeling architectures use attention to build generative models for language based tasks. Some of these models use all the available sequence tokens to generate an attention distribution which results in time complexity of $O(n^2)$. Alternatively, they utilize depthwise convolutions with softmax normalized kernels of size k acting as a limited-window self-attention, resulting in time complexity of $O(k \cdot n)$. In this paper, we introduce Time-aware Large Kernel (TaLK) Convolutions, a novel adaptive convolution operation that learns to predict the size of a summation kernel instead of using a fixed-sized kernel matrix. This method yields a time complexity of $O(n)$, effectively making the sequence encoding process linear to the number of tokens. We evaluate the proposed method on large-scale standard machine translation, abstractive summarization and language modeling datasets and show that TaLK Convolutions constitute an efficient improvement over other attention/convolution based approaches.

1. Introduction

Sequence modeling has seen some great breakthroughs through recent years with the introduction of the use of neural networks. Recurrent neural network methods (Sutskever et al., 2014; Bahdanau et al., 2015; Wu et al., 2016), convolution methods (Kim, 2014; Kalchbrenner et al., 2014; 2016; Gehring et al., 2017; Wu et al., 2019), and self-attention approaches (Paulus et al., 2018; Vaswani et al., 2017; Dai et al., 2019; Kitaev et al., 2020) have all yielded state-of-the-art results in various NLP tasks such as neural machine translation (NMT) (Sutskever et al., 2014; Wu et al., 2016; Britz et al., 2017; Aharoni et al., 2019), language modeling (Sundermeyer et al., 2012; Tran et al., 2016; Devlin

et al., 2019; Radford et al., 2019), automatic summarization (Paulus et al., 2018; Fan et al., 2018; Celikyilmaz et al., 2018), named entity recognition (Lample et al., 2016; Devlin et al., 2019) and sentiment analysis (Xu et al., 2016; Sachan et al., 2019).

Seemingly all modern approaches of sequence encoding rely on the use of attention to “filter” the excessive information given at a current time-step. Attention can be expressed as the weighted sum over context representations using attention weights that are usually generated from the context representations (self-attention) (Cheng et al., 2016). The transformer network (Vaswani et al., 2017) assigns attention weights for a given time-step to all available context token representations, while the newly proposed dynamic convolution (Wu et al., 2019) only computes an attention over a fixed context window.

Self-attention over all context tokens is computationally very expensive. Specifically, the transformer network has a time complexity of $O(n^2)$ where n is the length of the input sequence. Thus, modeling long-range dependencies becomes very challenging and the practicality of the self-attention method has been questioned. The more recent approach of dynamic convolutions (Wu et al., 2019) successfully reduced the time complexity to $O(k \cdot n)$ where k is the kernel size specified for each layer.

In this paper, we introduce a novel type of adaptive convolution, Time-aware Large Kernel (TaLK) convolutions, that learns the kernel size of a summation kernel for each time-step instead of learning the kernel weights as in a typical convolution operation. For each time-step, a function is responsible for predicting the appropriate size of neighbor representations to use in the form of left and right offsets relative to the time-step. The result is an efficient encoding method that reduces the time complexity to $O(n)$ and uses fewer parameters than all other methods. The method employs the fast Parallel Prefix Sum (Ladner & Fischer, 1980; Vishkin, 2003) operation which has a time complexity of $O(\log(n))$ to compute the *integral image* (Lewis, 1994), also known as *summed-area table* in the Computer Vision literature. This needs to be computed only once and can be used to calculate any summation between two boundary tokens in $O(1)$. Applying it on a sequence with length n only needs $O(n)$ time. To summarize, the contributions of

¹School of Computer Science, Carleton University, Canada. Correspondence to: Vasileios Lioutas <contact@vlioutas.com>.

this work are three-fold:

- We introduce a novel adaptive convolution based on summation kernel for sequence encoding.
- We show both analytically and empirically that the proposed kernel method has a smaller time complexity; it is faster than previous state-of-the-art approaches and is able to encode longer sentences quicker and with a smaller running memory footprint.
- We evaluate our method on three NLP tasks, machine translation, abstractive summarization and language modeling. We show that the proposed method can get comparative performance with previous methods on WMT En-De and WMT En-Fr benchmarks in machine translation, and set a new state-of-the-art result on the IWSLT De-En and CNN-DailyMail datasets, while in language modeling our method is able to perform comparatively with self-attention and outperform dynamic convolutions on the WikiText-103 benchmark dataset.

Our code and pre-trained models are available at github.com/lioutasb/TaLKConvolutions.

2. Related Work

In this section, we provide a brief review over various related sequence modeling methods, and related methods that enlarge the receptive field of a convolution operation.

2.1. Sequence Modeling

Sequence modeling is an important task in machine learning. An effective system should be able to comprehend and generate sequences similar to real data. Traditional approaches typically rely on the use of various kinds of recurrent neural networks such as long-short term memory networks (Hochreiter & Schmidhuber, 1997; Sutskever et al., 2014; Li et al., 2016; 2018) and gated recurrent unit networks (Cho et al., 2014; Nabil et al., 2016). These recurrent approaches are auto-regressive, which slows the process down for long sequences since they linearly depend on their own previous output tokens. Recent work is focused on exploring convolutional neural networks (CNN) methods (Kalchbrenner et al., 2016; Gehring et al., 2017; Wu et al., 2019) or self-attention methods (Vaswani et al., 2017; Zhang et al., 2018; Dai et al., 2019; Kitaev et al., 2020) which both facilitate the parallelization of the encoding process. In addition, since they are not auto-regressive, they allow the encoding process to capture stronger global and local dependencies.

Recently, Wu et al. (2019) proposed an alternative method to the original self-attention approach. Their window-based attention method with window size k can perform comparatively with self-attention modules that have access to all

available tokens at each time-step. They utilize a depthwise convolution with a generated softmax normalized kernel of size k for every time-step. This brings down the time complexity to $O(k \cdot n)$ from the quadratic complexity of the original self-attention model. However, this method has the drawback of being memory intensive for long sequences depending on the implementation used. Moreover, supporting larger kernel sizes can have a negative impact to running time. In another work, Shen et al. (2018) proposed computing intra-block self-attention weights within blocks of the input sequence and inter-block attention between all blocks to reduce the running memory of the full self-attention approach.

Our method differs from all these previous approaches in two main aspects. Specifically, instead of having all (or some) tokens available and then deciding which ones are needed to encode a time-step, we start from the current time-step representation and try to expand to the neighbor tokens in an adaptive manner. Additionally, instead of using attention for filtering the tokens used for encoding a time-step, we use all the information available in an adaptively decided window by utilizing a summation convolution kernel with summed-area tables, which improves upon previously proposed methodology by allowing us to reduce the complexity to $O(n)$, to produce a smaller running memory footprint, and to use less parameters than all other methods.

2.2. Dynamically Sized Receptive Field

Increasing the receptive field of a convolution layer without adding a computation overhead is a challenging task. By making deeper CNN models, we may be able to accumulate many fixed-sized receptive fields, however this comes at the cost of high computational demands. Nevertheless, this approach is shown to be successful in multiple state-of-the-art vision models (He et al., 2016; Szegedy et al., 2016). The overhead issue is often mitigated using a form of down-sampling, either via pooling layers (Lecun et al., 1998) or strided convolutions (Springenberg et al., 2015). Yu & Koltun (2016) proposed dilated convolutions, a method for enlarging the convolution kernel size by skipping intermediate pixels and thus, requiring less `multadds` operations.

The first work that suggested the use of learnable sized convolution kernels was box convolutions (Burkov & Lempitsky, 2018). The idea of using box filters with summed-area tables (Crow, 1984), commonly known as integral images dates back many years and it is well-known to the Computer Vision community, as it became particularly popular with the work of Viola & Jones (2001) in object detection. The summed-area table can be efficiently parallelized using the Parallel Prefix Sum method (Ladner & Fischer, 1980). This operation can be further accelerated as a hardware functional unit dedicated to compute the multi-parameter prefix-sum

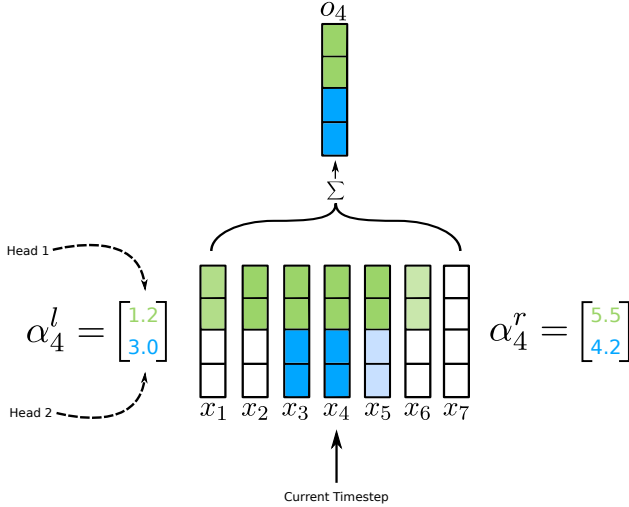


Figure 1. The Time-aware Large Kernel convolution operation. For the current time-step, we compute the left and right offsets for each head, and then sum all the representation vectors inside these boundaries. This operation can be efficiently computed using summed-area tables with time complexity $O(\log(n))$ and compute the output representation for each time-step in $O(n)$ time.

operation (Vishkin, 2003).

Burkov & Lempitsky (2018) method is optimizing the kernel size parameters using approximate gradients by normalizing the sum by the area of the box. Zhang et al. (2019) extended this idea by using interpolation to exploit non-integer coordinates. Inspired by this idea, we develop the proposed method for one-dimensional case of sequences. In contrast to the two previous methods, instead of using a fixed number of learnable sized kernels, we adaptively condition the size of the kernel on each input representation, effectively generating a different kernel size for each time-step token.

3. Methodology

In this section, we present the proposed adaptive Time-aware Large Kernel (TaLK) Convolution method. First, we introduce the approach that computes a convolution operation using large kernels in $O(n)$ time, which assumes that left and right offsets are given. Next, we present our proposed method for generating offsets dynamically for each time-step. We then expand upon our method to use multiple heads and normalize the summed output vector. Next, we explain how to use TaLK Convolutions for decoding. Finally, we present the computational complexity analysis and comparison for the proposed method. Figure 1 illustrates the Time-aware Large Kernel Convolution operation for a specific time-step during encoding.

3.1. One-dimensional Large Kernel Convolution

Let $X = \{x_1, x_2, \dots, x_n\}$ denote an input sequence, where n is the length of the sequence, $x_i \in \mathbb{R}^d$ is the current input representation for the i -th word (i.e., the i -th time-step) and d denotes the dimensionality of the vector representation (i.e., the number of channels).

The goal of this paper is to reduce the encoding time complexity for sequence modeling to $O(n)$. In other words, we set out to make the encoding at each time-step independent of the size of the receptive field. In addition, we want to explore alternative methods to the successful self-attention mechanism by equally using the number of neighbor tokens to represent a time-step instead of generating an attention distribution over the tokens. Specifically, we assume that simply summing the appropriate number of token representations is enough to represent the current time-step. That is, we encode the representation at the i -th time-step by

$$o_i = \sum_{j=\alpha_i^l}^{\alpha_i^r} x_j, \quad (1)$$

where $1 \leq \alpha_i^l \leq i \leq \alpha_i^r \leq n$ are the lower (left offset) and upper (right offset) bounds of the kernel size.

Applying Equation 1 for each time-step i separately is inefficient since we do repetitive summations over the same representations. Zhang et al. (2019) showed that using the summed-area table (Crow, 1984), we can accelerate a summation convolution operation to any kernel size. Specifically, let $\mathcal{S} = \{\mathcal{S}_0, \mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_n\}$ be the summed-area table computed using

$$\begin{cases} \mathcal{S}_0 = 0, \\ \mathcal{S}_i = \mathcal{S}_{i-1} + x_i, \quad 1 \leq i \leq n. \end{cases} \quad (2)$$

Given the left offset α_i^l and the right offset α_i^r , we can compute the summation denoted as o_i of the features between these offsets using the summed-area table

$$o_i = \mathcal{S}_{\alpha_i^r} - \mathcal{S}_{\alpha_i^l-1} \quad (3)$$

3.2. Time-aware Large Kernel Generation

Given the one-dimensional large kernel convolution above, it is important to determine the left and right offsets for computing representations at each time-step. The key of the proposed method is an adaptive time-aware large kernel convolution operation which has kernel sizes that vary over time as a learned function of the individual time steps; that is, we propose to learn the offsets of the summation kernel above for each time-step.

Specifically, we propose to use a function $f^{\{l,r\}} : \mathbb{R}^d \rightarrow \mathbb{R}$ to generate for each x_i the left $\tilde{\alpha}_i^l$ and right $\tilde{\alpha}_i^r$ relative

offsets, where $\tilde{a}_i^{\{l,r\}} = \sigma(f^{\{l,r\}}(x_i)) \in [0, 1]$. For each $\tilde{a}_i^{\{l,r\}}$ relative offset, we convert it to the absolute offset counterpart in the following way

$$\begin{aligned} a_i^l &= i - \tilde{a}_i^l \cdot l_{\max} \\ a_i^r &= i + \tilde{a}_i^r \cdot r_{\max}, \end{aligned} \quad (4)$$

where $l_{\max} \in \mathbb{Z}_{\geq 0}$ is the maximum allowed tokens to the left and $r_{\max} \in \mathbb{Z}_{\geq 0}$ is the maximum allowed tokens to the right.

The absolute offsets up to this point represent real positive numbers. In the next step, we need to convert these numbers to integer indexes so we can select from the summed-area table using the Equation (3). Inspired by Zhang et al. (2019), we use one-dimensional interpolation to sample from the summed-area table by using the positive real-valued offsets a_i^l, a_i^r as follows

$$\begin{aligned} \mathcal{S}_{a_i^l-1} &= \gamma^l \cdot \mathcal{S}_{\lfloor a_i^l \rfloor - 1} + (1 - \gamma^l) \cdot \mathcal{S}_{\lceil a_i^l \rceil - 1}, \\ \mathcal{S}_{a_i^r} &= (1 - \gamma^r) \cdot \mathcal{S}_{\lfloor a_i^r \rfloor} + \gamma^r \cdot \mathcal{S}_{\lceil a_i^r \rceil}, \end{aligned} \quad (5)$$

where $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ are the floor and ceiling operators, $\gamma^l = \lceil a_i^l \rceil - a_i^l$ and $\gamma^r = a_i^r - \lfloor a_i^r \rfloor$. The above equation is continuous and differentiable in the interpolation neighborhood. The partial derivatives of $\mathcal{S}_{a_i^{\{l,r\}}}$ with respect to $\tilde{a}_i^{\{l,r\}}$ are given by

$$\begin{aligned} \frac{\partial \mathcal{S}_{a_i^l-1}}{\partial \tilde{a}_i^l} &= l_{\max} (\mathcal{S}_{\lfloor a_i^l \rfloor - 1} - \mathcal{S}_{\lceil a_i^l \rceil - 1}), \\ \frac{\partial \mathcal{S}_{a_i^r}}{\partial \tilde{a}_i^r} &= r_{\max} (\mathcal{S}_{\lceil a_i^r \rceil} - \mathcal{S}_{\lfloor a_i^r \rfloor}). \end{aligned} \quad (6)$$

The partial derivatives of $\mathcal{S}_{a_i^{\{l,r\}}}$ with respect to $\mathcal{S}_{\lfloor a_i^{\{l,r\}} \rfloor}$ and $\mathcal{S}_{\lceil a_i^{\{l,r\}} \rceil}$ tokens are given by

$$\begin{aligned} \frac{\partial \mathcal{S}_{a_i^l-1}}{\partial \mathcal{S}_{\lfloor a_i^l \rfloor - 1}} &= \gamma^l, & \frac{\partial \mathcal{S}_{a_i^l-1}}{\partial \mathcal{S}_{\lceil a_i^l \rceil - 1}} &= (1 - \gamma^l), \\ \frac{\partial \mathcal{S}_{a_i^r}}{\partial \mathcal{S}_{\lfloor a_i^r \rfloor}} &= (1 - \gamma^r), & \frac{\partial \mathcal{S}_{a_i^r}}{\partial \mathcal{S}_{\lceil a_i^r \rceil}} &= \gamma^r. \end{aligned} \quad (7)$$

3.3. Output Normalization and Offsets Dropout

The idea of summing all the features in a window of size $\lfloor a_i^l, a_i^r \rfloor$ works well for shallow models. However, as the representation vectors at different time-steps are computed from summations over different numbers of neighbors, their magnitudes of values can be different. As we introduce more layers, the disproportional magnitude of the inputs makes learning harder for the nodes in the layers that follow. To address this problem, we propose to normalize the output representations of TaLK Convolutions as follows

$$\tilde{o}_i = o_i \cdot \left(\frac{1}{l_{\max} + r_{\max} + 1} \right). \quad (8)$$

Such a simple window size based normalization can effectively get rid of the output magnitude differentiation problem resulted from summation kernels.

In addition, we regularize the predicted offsets $\tilde{a}_i^{\{l,r\}}$ using Dropout (Hinton et al., 2012; Srivastava et al., 2014). Specifically, during training we drop out every predicted offset with probability p . This helps to prevent the model from quickly optimizing towards a specific window size and be able to generate more diverse offsets.

3.4. Multi-headed Kernels

Although the offset computation above provides a mechanism that offers adaptive receptive fields for summation kernels at different time steps, a single pair of left and right offsets for all d dimensions cannot yield good results, as different features might be related to their counterpart in the neighbor tokens in different way. Inspired by the idea of multi-head attention (Vaswani et al., 2017; Wu et al., 2019), we further propose to extend our proposed convolution kernel into a multi-head version by allowing different representation features, i.e., channels, to have different left and right offsets for each time-step. Moreover, instead of having entirely different convolution offsets across multiple channels, we adopt a depthwise version by separating the feature channels into multiple groups, each of which share the same pair of left and right offsets.

Specifically, we tie every subsequent number of $R = \frac{d}{H}$ channels together and group the channels into H groups for each x_i , where H is the number of heads. This results to $\hat{X} = \{\hat{x}_1, \hat{x}_2, \dots, \hat{x}_n\}$, where $\hat{x}_i \in \mathbb{R}^{H \times R}$. Then we use a function $f^{\{l,r\}} : \mathbb{R}^{H \times R} \rightarrow \mathbb{R}^H$ to generate for each \hat{x}_i a vector of H left relative offsets $\tilde{\alpha}_i^l$ or right relative offsets $\tilde{\alpha}_i^r$ via $\tilde{\alpha}_i^{\{l,r\}} = \sigma(f^{\{l,r\}}(\hat{x}_i)) \in [0, 1]^H$.

3.5. Decoding Using TaLK Convolutions

In an encoder/decoder sequence generation scheme (Sutskever et al., 2014), the encoder part of the model has access to both past and future tokens. The decoding part, however, must have access only to past tokens that are generated so far. Enforcing this with TaLK Convolutions is straightforward by setting the r_{\max} value to zero.

3.6. Module Architecture and Implementation

For sequence modeling, we follow a similar module architecture as described in Wu et al. (2019). Specifically, we apply a linear layer to project the input embedding tokens from d to $2d$ and then we apply a gated linear unit (GLU) (Dauphin et al., 2017). Next, we apply the TaLK Convolution operation as described in Section 3.2. Finally, we apply a projection layer to the output representations from TaLK Convolution with size $W \in \mathbb{R}^{d \times d}$. Figure 2 illustrates the

Table 1. Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. n is the sequence length, d is the representation dimension, k is the kernel size of convolutions and $n_{buckets}$ is the number of hash buckets.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Recurrent (Sutskever et al., 2014)	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional (Kalchbrenner et al., 2016; Gehring et al., 2017)	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(\log_k(n))$ or $O(n/k)$
Self-Attention (Vaswani et al., 2017)	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Dynamic Convolutions (Wu et al., 2019)	$O(k \cdot n \cdot d)$	$O(1)$	$O(n/k)$
Reformer (Kitaev et al., 2020)	$O(n \cdot \log(n) \cdot d)$	$O(\log(n))$	$O(n/n_{buckets})$
TaLK Convolutions (Ours)	$O(n \cdot d)$	$O(\log(n))$	$O(n/(l_{max} + r_{max} + 1))$

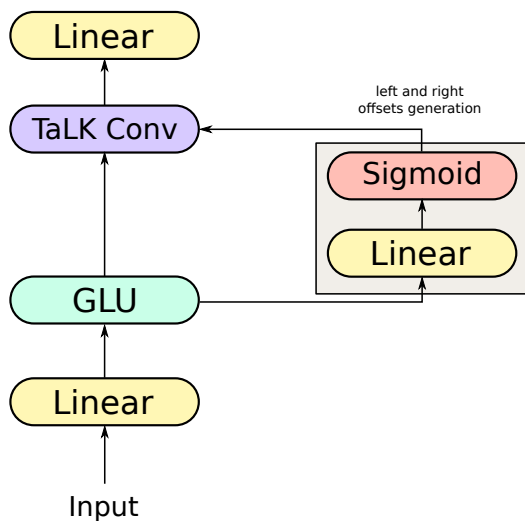


Figure 2. The proposed TaLK Convolution unit.

TaLK Convolution unit. In addition, we substitute all ReLU activation functions with the Swish function (Ramachandran et al., 2017). Similar to (Vaswani et al., 2017; Wu et al., 2019), the overall architecture is composed by the TaLK Convolution unit which replaces the self-attention or the dynamic convolution unit followed by the position-wise feed-forward network (FFN) unit.

The summed-area table (Equation 2) can be efficiently computed on a GPU by performing a fast Parallel Prefix Sum (Ladner & Fischer, 1980) over the token dimension. This operation is usually efficiently implemented on modern deep learning frameworks under the name of cumulative sum. Applying the relative offsets to the summed-area table using core functions from deep learning frameworks is not a trivial task. Such an implementation is usually very inefficient leading to slower computation and memory overhead. For this reason, we implemented the operation using CUDA kernels that enabled us to parallelize the computation for each token.

3.7. Computational Complexity

In this section we compare the complexity of the TaLK Convolution operation against different modules for encoding an input sequence of representations. This comparison is shown on Table 1. We follow a similar comparison as analyzed by Vaswani et al. (2017). Our comparison is based on three criteria: the time complexity of the operation, the amount of computations that can be executed in parallel and the path length between long-range dependencies.

As shown in Table 1, our proposed method requires the least number of operations. Specifically, it has a linear time complexity to encode a sequence and does not depend on hyper-parameter decisions such as the kernel size. In terms of the number of computations that can be parallelized, our method needs logarithmic time to compute the summed-area table (Equation 2). It is true that our method does not have a constant number of sequentially executed operations like all the other non-autoregressive counterpart methods, but the logarithmic time our method is requiring is inexpensive to compute even for very long sequences.

It is shown by Kolen & Kremer (2001) that a short path between any combination of token positions in the input and output sequences makes it easier to learn long-range dependencies. In practice, doubts have been cast over the ability of self-attention to model long-range dependencies (Tang et al., 2018; Wu et al., 2019). Wu et al. (2019) showed that using a limited context window can outperform self-attention. Our method has the advantage that the number of required computations is independent of the maximum window size and thus, it can be tuned or learned without extra cost.

4. Experiments

4.1. Datasets and Evaluation

We evaluated our proposed encoding technique on machine translation, abstractive summarization and language mod-

Table 2. Machine translation accuracy in terms of BLEU for WMT En-De and WMT En-Fr on newstest2014.

Model	Param (En-De)	WMT En-De	WMT En-Fr
Gehring et al. (2017)	216M	25.2	40.5
Vaswani et al. (2017)	213M	28.4	41.0
Ahmed et al. (2017)	213M	28.9	41.4
Chen et al. (2018)	379M	28.5	41.0
Shaw et al. (2018)	-	29.2	41.5
Ott et al. (2018)	210M	29.3	43.2
Wu et al. (2019)	213M	29.7	43.2
Kitaev et al. (2020)	213M	29.1	-
TaLK Convolution (Ours)	209M	29.6	43.2

Table 3. Machine translation accuracy in terms of BLEU on IWSLT De-En.

Model	Param	IWSLT De-En
Deng et al. (2018)	-	33.1
Vaswani et al. (2017)	47M	34.4
Wu et al. (2019)	43M	35.2
TaLK Convolution (Ours)	42M	35.5

eling. These three tasks are considered touchstone and challenging in the NLP field.

Machine Translation On the machine translation task, we report results on three mainstream benchmark datasets: WMT English to German (En-De), WMT English to French (En-Fr) and IWSLT German to English (De-En).

For all datasets, we replicated the pre-processing steps mentioned in Wu et al. (2019). Specifically, for the WMT En-De we used the WMT’16 training data that consists of 4.5M sentence pairs. We validated on newstest2013 and tested on newstest2014. We employed byte-pair encoding (BPE) (Sennrich et al., 2016) to the sentences, with a 32K joint source and target vocabulary. For the WMT En-Fr, we used 36M training sentence pairs from WMT’14. We validated on newstest2012+2013 and tested on newstest2014 evaluation datasets. Using BPE, we generated a joint vocabulary between the source and the target languages of size 40K tokens. The IWSLT De-En consists of 160K training sentence pairs. We lower cased all sentences and used a 10K joint BPE vocabulary.

For all datasets, we measured case-sensitive tokenized BLEU scores using `multi-bleu`¹. Similarly to Vaswani

¹<https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi%2Dbleu.perl>

et al. (2017), we applied compound splitting for WMT En-De. We trained five random initializations of each model configuration and report test accuracy of the seed which resulted in the highest validation BLEU score. For all datasets, we used beam search with beam width 5. Similar to Wu et al. (2019), we tuned a length penalty as well as the number of checkpoints to average on the validation set.

Abstractive Summarization For the abstractive summarization task, we decided to experiment with the CNN-DailyMail (Hermann et al., 2015; Nallapati et al., 2016) dataset. The dataset is composed by approximately 280K news articles with associated multi-sentence summaries. We followed the same pre-processing steps as described by Wu et al. (2019). The BPE vocabulary consists of 30K subword tokens. We report results using the F1-Rouge (Rouge-1, Rouge-2 and Rouge-L) metric (Lin, 2004). For generating the summaries, similar to Wu et al. (2019) we tuned the maximum output length, disallowing repeating the same trigram, and we apply a stepwise length penalty.

Language Modeling We experimented on the WikiText-103 (Merity et al., 2017) benchmark dataset. The training data contains approximately 100M tokens. A vocabulary of about 260K tokens was used, by discarding all tokens with a frequency below 3 as described in Merity et al. (2017). We followed Baevski & Auli (2019) and applied adaptive input representations. We replicated their setup and partition the training data into blocks of 512 contiguous tokens while ignoring document boundaries.

4.2. Experiment Details

Hyper-Parameters For the machine translation models, we followed the same hyper-parameter setup as described in Wu et al. (2019). Specifically, we follow for WMT En-De and WMT En-Fr datasets the model hidden size d was set to 1024, the feed-forward hidden size d_{ff} was set to 4096 and the number of layers for the encoder and the decoder

Table 4. Results on CNN-DailyMail abstractive summarization.

Model	Param	Rouge-1	Rouge-2	Rouge-L
LSTM (Paulus et al., 2018)	-	38.30	14.81	35.49
CNN (Fan et al., 2018)	-	39.06	15.38	35.77
Self-Attention Baseline (Wu et al., 2019)	90M	39.26	15.98	36.35
Lightweight Convolution (Wu et al., 2019)	86M	39.52	15.97	36.51
Dynamic Convolution (Wu et al., 2019)	87M	39.84	16.25	36.73
TaLK Convolution (Standard)	59M	40.03	18.45	36.13
TaLK Convolution (Deep)	83M	40.59	18.97	36.81

Table 5. Test perplexity on WikiText-103. We used adaptive inputs similar to Baevski & Auli (2019) and show that our method yields better perplexity than dynamic convolutions and comparative performance with self-attention.

	Param	Test
Grave et al. (2017)	-	40.8
Dauphin et al. (2017)	229M	37.2
Merity et al. (2018)	151M	33.0
Rae et al. (2018)	-	29.2
Baevski & Auli (2019)	247M	20.5
Dynamic Convolutions	255M	25.0
TaLK Convolution (Ours)	240M	23.3

was set to 7 and 6 respectively. The number of heads was set to 16 and the l_{\max}, r_{\max} values to 3, 7, 15, 31×4 for each layer. For IWSLT De-En, the model hidden size d was set to 512, the feed-forward hidden size d_{ff} was set to 1024 and the number of layers for the encoder and the decoder was set to 7 and 6 respectively. The number of heads was set to 4 and the l_{\max}, r_{\max} values to 1, 3, 7, 15×4 for each layer.

For the abstractive summarization models, we tested our method on two types of model configurations, the Standard and the Deep configurations. Both settings have a hidden size d of 512, a feed-forward hidden size d_{ff} of 2048 and number of heads to be 8. The Standard model has 7 encoder and 6 decoder layers with the l_{\max}, r_{\max} values be 3, 7, 15, 31×4 for each layer. The Deep model has 10 layers for both the encoder and decoder with the l_{\max} values be 3, 7, 15, 31×7 and the r_{\max} be 3, 7, 31×8 .

For the language model, we followed the same configuration as Baevski & Auli (2019). We used 17 decoding layers, each layer with a 1024 hidden size, a 4096 feed-forward hidden size and 8 heads. The adaptive input factor was set to 4. The l_{\max} values were set to 3, 7, 15, 31, 63×12 and r_{\max} to zero for each layer.

Optimization We used the Adam optimizer (Kingma & Ba, 2015) with default values. In addition, our models were optimized using the cosine learning rate schedule (Loshchilov & Hutter, 2017). We linearly warmed up for 10K steps from 10^{-7} to 10^{-3} . For IWSLT De-En, we used the inverse square root learning rate schedule (Vaswani et al., 2017). We set the dropout to 0.3 for WMT En-De and IWSLT De-En and 0.1 for WMT En-Fr.

For the WMT En-De and WMT En-Fr benchmarks, the batch size was set to 3,584 tokens per batch, per GPU. We accumulated the gradients for 16 batches before applying an update which results in an effective batch size of 450K tokens. We trained the WMT En-De model for 30K steps and the WMT En-Fr for 80K steps. For IWSLT De-En, we trained on a single GPU with 4,000 maximum tokens per batch for 50K steps. For the abstractive summarization models we followed the same setup as in Wu et al. (2019) and for the language model the same setup as in Baevski & Auli (2019).

Hardware Details We trained the WMT En-De, WMT En-Fr, CNN-DailyMail and WikiText-103 models on 8 NVIDIA RTX 2080 Ti GPUs using mixed-precision training (Micikevicius et al., 2018) and the IWSLT De-En model using a single GPU. We employed our own CUDA implementation, wrapped as a standalone PyTorch layer for the TaLK Convolution operation. All experiments were run using the Fairseq (Ott et al., 2019) toolkit.

4.3. Results on Machine Translation

We demonstrate the effectiveness of our model in the WMT En-De and WMT En-Fr translation benchmarks. Table 2 shows that our method is able to achieve comparable results to current state-of-the-art methods. Specifically, our method was able to match the state-of-the-art score on WMT En-Fr, a benchmark dataset that is considered indicative for the effectiveness of a method due to the large number of training examples (36M) it contains. Additionally for WMT En-De, our method is only 0.1 BLEU points behind the current state-

Table 6. Throughput and memory consumption decrease measured for different sequence lengths (n) on a batch of size 10 with each token being represented with $d = 1024$ and $H = 16$. Throughput is calculated across 100K iterations of a single input encoding execution for each method. Memory decrease is computed as how many times less memory we need to encoding the input embedding compared to Self-Attention. Larger numbers indicate better performance.

Method	$n = 10$		$n = 100$		$n = 1,000$		$n = 10,000$	
	iter/sec	Mem. ↓	iter/sec	Mem. ↓	iter/sec	Mem. ↓	iter/sec	Mem. ↓
Self-Attention	4576	1x	3437	1x	102	1x	OOM	1x
DynamicConv ($k = 3$)	3739	1x	3308	0.99x	443	2.8x	45	25.4x
DynamicConv ($k = 31$)	4535	0.97x	3860	1x	325	2.7x	29	20.2x
TaLK Convolution	9686	1.1x	6126	1.1x	898	3.1x	92	26.4x

of-the-art score. It is important to underline, however, that our method uses the least number of parameters compared to the other counterpart methods.

Table 3 shows results for IWSLT De-En benchmark dataset. Following Wu et al. (2019), we employed a smaller model with less parameters to reflect the size of the dataset. Specifically, we set d to 512, d_{ff} to 1024 and H to 4. Furthermore, we disabled the GLU unit that is described in Section 3.6 and made the input projection layer to size $W \in \mathbb{R}^{d \times d}$. Our method was able to outperform all other methods setting a new state-of-the-art result.

4.4. Results on Abstractive Summarization

We evaluated the proposed method on the task of abstractive summarization. We test the method’s ability to process long documents on the CNN-DailyMail dataset. We encode an article of up to 400 sub-words and we generate a summarization composed from multiple sentences. Table 4 shows the results of our experiments. Our Standard model is able to achieve better results on the Rouge-1 and Rouge-2 metrics than previous methods. In addition, the Standard model is using significantly less parameters, approximately 30M parameters less. The Deep model uses more layers to closely match the number of parameters and is able to outperform all previous models. This shows that our method is able to encode long sequences successfully without having the need to have access to all context.

4.5. Results on Language Modeling

We evaluated our method on the task of language modeling. We considered the WikiText-103 benchmark dataset and we compared against recent methods in the literature. Particularly, we followed the setup that was implemented in the adaptive inputs baseline (Baevski & Auli, 2019). This work suggest the use of self-attention with adaptive input representations. We substituted the self-attention module with our method. In order to assimilate the number of parameters used in their experiments, we increased the number

of layers by one. As seen on Table 5, our method yields better perplexity than dynamic convolutions when trained using the same settings, including the same maximum kernel size. In addition, we get comparative performance with self-attention. Moreover, we use less number of parameters than the best comparison methods.

4.6. Encoding Inference Speed Comparison

We also compared our method against other non-autoregressive methods in terms of encoding inference speed and memory consumption. We measured the speed using a single NVIDIA RTX 2080 Ti GPU with full precision floating-point arithmetic (FP32). Specifically, we measured the throughput of encoding a batch of size $B = 10$, $d = 1024$ and $H = 16$. For each method, we only took into consideration the time it takes to process using the core approach of each encoding method.

For self-attention (Vaswani et al., 2017), we only timed the attention operation $\text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$. For dynamic convolutions (Wu et al., 2019), we only timed the operation $\text{DepthwiseConv}(X, \text{softmax}(W_{\text{dyn}}, i, c))$ where $W_{\text{dyn}} \in \mathbb{R}^{n \cdot B \cdot H \times K}$ is the generated kernel for each time-step. The authors of dynamic convolutions proposed two ways of implementing their method. The first method uses the standard convolution unfolding function which is faster for longer sequences. The second approach is the band matrix trick method which copies and expands the normalized weights matrix into a band matrix. This second approach yields faster execution time for shorter sequences but is more memory intensive. In order to be fair, in our experiments we used unfolding to sequences longer than 500 tokens and band matrices for shorter sequences. We also set K to 3 and 31, the first being the smallest kernel size dynamic convolutions use and the second being the largest. Finally, for our method we measured the time to compute the large kernel convolution operation given the relative offsets. We evaluated for 100K iterations across four different sequence lengths n .

Table 6 shows that our method yields much better through-

Table 7. Ablation on IWSLT De-En validation set. (+) indicates that a result includes all preceding features.

Model	Param	BLEU
TaLK Convolution ($a_i^l, a_i^r=1 \times 7, H=1$)	42M	diverges
+ Output Normalization	42M	35.70 ± 0.1
+ Increasing Max Offsets ($a_i^l, a_i^r=1, 3, 7, 15 \times 4$)	42M	36.23 ± 0.1
+ Offsets Dropout ($p=0.1$)	42M	36.37 ± 0.05
+ Fully-headed Kernels ($H=512$)	47M	36.51 ± 0.07
+ Multi-headed Kernels ($H=4$)	42M	36.65 ± 0.05
Replacing Swish with ReLU	42M	36.21 ± 0.05

put than all other methods. Specifically, the number of iterations of self-attention per second is comparable to dynamic convolutions for short sentences ($n < 500$). Our method allows for more sentences to be processed each second, leading to a much higher throughput. For longer sentences, self-attention is notably slower than our method and for the case of $n = 10,000$, self-attention was running out-of-memory and was not able to execute an iteration. Although our method has a logarithmic time complexity for computing the summed-area table (Section 3.7), the fact that we are computing a much more inexpensive in terms of complexity operation, specifically we only utilize additions, other methods with $O(1)$ complexity are less efficient due to the employment of both multiplication as well as addition operations. Therefore, our method has a considerably higher throughput compared to dynamic convolutions.

Furthermore, we examined the running memory requirements for all three different non-autoregressive methods. We compared dynamic convolutions and our proposed method against self-attention and report the number of times we reduced the running memory compared to self-attention. For all sequence length cases, our method requires less memory than dynamic convolutions when compared to the “expensive” self-attention operation. The times we were able to decrease the memory consumption can be seen on Table 6.

4.7. Model Ablation

In order to evaluate the importance of the different choices for the TaLK Convolutions, we varied our baseline model, described in Section 3.2, using the different proposed extensions mentioned in Sections 3.3 and 3.4. We measured the performance on the validation set of the IWSLT De-En translation benchmark dataset. We used beam search as described in Section 4.1. We report the results in Table 7.

Initially, we modified the baseline model with the addition of the output normalization (Section 3.3). As seen in Table 7, the original method is not able to converge. This validates our intuition that since we are summing the available information inside the kernel, not normalized outputs make

learning difficult for the layers that follow. Next, we increased the values l_{\max}, r_{\max} to allow larger adaptive kernel sizes which yielded a higher performance without additional computation cost. Further, we introduced a dropout unit with probability $p = 0.1$ on the generated relative offsets. This allowed for the performance to increase further as we stopped the model from overfitting over the same window size. Next, we increased the number of heads H from 1 to 512 (all available dimensions) and we called this fully-head TaLK Convolution. We can see that by treating each of the 512 dimensions separately and generating 512 relative offsets, we were able to increase the performance. However, we believe that by having each dimension generate its own offsets actually brings some noise. Thus, we reduced the number of heads to $H = 4$ which increased the performance even more. Finally, we show that by substituting the Swish activation function with the ReLU function the performance drops which justifies our decision to use the former.

5. Conclusion

In this work, we presented Time-aware Large Kernel (TaLK) Convolutions, a novel adaptive convolution method based on summation kernel for sequence representation and encoding. It learns to predict the kernel boundaries for each time-step of the sequence. In contrast to all other non-autoregressive methods, this approach needs true linear time $O(n)$ with respect to the sequence length, while being able to successfully encode a sequence without using the notion of attention. We validated the proposed method on three NLP tasks, machine translation, abstractive summarization and language modeling, and achieved a comparative performance. Moreover, we showed both analytically and empirically that the proposed method is faster than previous approaches and that it is able to encode longer sentences quicker and with a smaller running memory footprint. For future work, we plan to apply our method to other sequential tasks such as question answering and the PG-19 benchmark dataset (Rae et al., 2020). We will also explore this novel convolution mechanism in the area of computer vision.

Acknowledgements

This research was supported by the Canada Research Chairs program and the NSERC Discovery grant. We would like to express our gratitude to our anonymous reviewers for their valuable comments and feedback. A special thank you to Vasileia Karasavva for editing and proofreading the final manuscript.

References

- Aharoni, R., Johnson, M., and Firat, O. Massively multilingual neural machine translation. 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)*, 2019.
- Ahmed, K., Keskar, N. S., and Socher, R. Weighted transformer network for machine translation, 2017. URL <https://arxiv.org/abs/1711.02132>.
- Baevski, A. and Auli, M. Adaptive input representations for neural language modeling. In *International Conference on Learning Representations*, 2019.
- Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*, 2015.
- Britz, D., Goldie, A., Luong, M.-T., and Le, Q. Massive exploration of neural machine translation architectures. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017.
- Burkov, E. and Lempitsky, V. Deep neural networks with box convolutions. In *Advances in Neural Information Processing Systems*. 2018.
- Celikyilmaz, A., Bosselut, A., He, X., and Choi, Y. Deep communicating agents for abstractive summarization. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018.
- Chen, M. X., Firat, O., Bapna, A., Johnson, M., Macherey, W., Foster, G., Jones, L., Schuster, M., Shazeer, N., Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, L., Chen, Z., Wu, Y., and Hughes, M. The best of both worlds: Combining recent advances in neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018.
- Cheng, J., Dong, L., and Lapata, M. Long short-term memory-networks for machine reading. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016.
- Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.
- Crow, F. C. Summed-area tables for texture mapping. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, 1984.
- Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q., and Salakhutdinov, R. Transformer-xl: Attentive language models beyond a fixed-length context. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.
- Dauphin, Y. N., Fan, A., Auli, M., and Grangier, D. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, 2017.
- Deng, Y., Kim, Y., Chiu, J., Guo, D., and Rush, A. M. Latent alignment and variational attention. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2018.
- Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.
- Fan, A., Grangier, D., and Auli, M. Controllable abstractive summarization. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*. Association for Computational Linguistics, 2018.
- Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. Convolutional sequence to sequence learning. In *ICML*, 2017.
- Grave, E., Joulin, A., and Usunier, N. Improving neural language models with a continuous cache. In *International Conference on Learning Representations*, 2017.
- He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- Hermann, K. M., Kočiský, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., and Blunsom, P. Teaching machines to read and comprehend. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, 2015.

- Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. Improving neural networks by preventing co-adaptation of feature detectors, 2012. URL <https://arxiv.org/abs/1207.0580>.
- Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural Computation*, 1997.
- Kalchbrenner, N., Grefenstette, E., and Blunsom, P. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2014.
- Kalchbrenner, N., Espeholt, L., Simonyan, K., van den Oord, A., Graves, A., and Kavukcuoglu, K. Neural machine translation in linear time, 2016. URL <https://arxiv.org/abs/1610.10099>.
- Kim, Y. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2014.
- Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.
- Kitaev, N., Kaiser, L., and Levskaya, A. Reformer: The efficient transformer. In *International Conference on Learning Representations*, 2020.
- Kolen, J. F. and Kremer, S. C. *Gradient Flow in Recurrent Nets: The Difficulty of Learning Long-Term Dependencies*. IEEE, 2001.
- Ladner, R. E. and Fischer, M. J. Parallel prefix computation. *J. ACM*, 1980.
- Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., and Dyer, C. Neural architectures for named entity recognition. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016.
- Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.
- Lewis, J. Fast template matching. *Vis. Interface*, 1994.
- Li, J., Galley, M., Brockett, C., Spithourakis, G., Gao, J., and Dolan, B. A persona-based neural conversation model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016.
- Li, Y., Pan, Q., Wang, S., Yang, T., and Cambria, E. A generative model for category text generation. *Information Sciences*, 2018.
- Lin, C.-Y. ROUGE: A package for automatic evaluation of summaries. In *Association for Computational Linguistics*, 2004.
- Loshchilov, I. and Hutter, F. Sgdr: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017.
- Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017.
- Merity, S., Keskar, N. S., and Socher, R. An analysis of neural language modeling at multiple scales, 2018. URL <http://arxiv.org/abs/1803.08240>.
- Micikevicius, P., Narang, S., Alben, J., Diamos, G., Elsen, E., Garcia, D., Ginsburg, B., Houston, M., Kuchaiev, O., Venkatesh, G., and Wu, H. Mixed precision training. *International Conference on Learning Representations*, 2018.
- Nabil, M., Atyia, A., and Aly, M. CUFÉ at SemEval-2016 task 4: A gated recurrent model for sentiment classification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, 2016.
- Nallapati, R., Zhou, B., dos Santos, C., Gülçehre, Ç., and Xiang, B. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, 2016.
- Ott, M., Edunov, S., Grangier, D., and Auli, M. Scaling neural machine translation. *Proceedings of the Third Conference on Machine Translation: Research Papers*, 2018.
- Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., Ng, N., Grangier, D., and Auli, M. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.
- Paulus, R., Xiong, C., and Socher, R. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations*, 2018.
- Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners, 2019. URL <https://d4mucfpxsywv.cloudfront.net/better-language-models/language-models.pdf>.

- Rae, J. W., Dyer, C., Dayan, P., and Lillicrap, T. P. Fast parametric learning with activation memorization. In *ICML*, 2018.
- Rae, J. W., Potapenko, A., Jayakumar, S. M., Hillier, C., and Lillicrap, T. P. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations*, 2020.
- Ramachandran, P., Zoph, B., and Le, Q. V. Searching for activation functions, 2017. URL <https://arxiv.org/abs/1710.05941>.
- Sachan, D. S., Zaheer, M., and Salakhutdinov, R. Revisiting lstm networks for semi-supervised text classification via mixed objective function. In *AAAI*, 2019.
- Sennrich, R., Haddow, B., and Birch, A. 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)*, 2016.
- Shaw, P., Uszkoreit, J., and Vaswani, A. Self-attention with relative position representations. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, 2018.
- Shen, T., Zhou, T., Long, G., Jiang, J., and Zhang, C. Bi-directional block self-attention for fast and memory-efficient sequence modeling. In *International Conference on Learning Representations*, 2018.
- Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. Striving for simplicity: The all convolutional net. *International Conference on Learning Representations*, 2015.
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 2014.
- Sundermeyer, M., Schlüter, R., and Ney, H. Lstm neural networks for language modeling. In *INTERSPEECH*, 2012.
- Sutskever, I., Vinyals, O., and Le, Q. V. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, 2014.
- Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, 2016.
- Tang, G., Müller, M., Rios, A., and Sennrich, R. Why self-attention? a targeted evaluation of neural machine translation architectures. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.
- Tran, K., Bisazza, A., and Monz, C. Recurrent memory networks for language modeling. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 2016.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.
- Viola, P. and Jones, M. Robust real-time object detection. In *International Journal of Computer Vision*, 2001.
- Vishkin, U. Prefix sums and an application thereof. : 09/224,104, 2003/04/01/ 2003. URL <http://www.google.com/patents?id=qCAPAAAAEBAJ>.
- Wu, F., Fan, A., Baevski, A., Dauphin, Y., and Auli, M. Pay less attention with lightweight and dynamic convolutions. In *International Conference on Learning Representations*, 2019.
- Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Łukasz Kaiser, Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., and Dean, J. Google’s neural machine translation system: Bridging the gap between human and machine translation, 2016. URL <https://arxiv.org/abs/1609.08144>.
- Xu, J., Chen, D., Qiu, X., and Huang, X. Cached long short-term memory neural networks for document-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016.
- Yu, F. and Koltun, V. Multi-scale context aggregation by dilated convolutions. *International Conference on Learning Representations*, 2016.
- Zhang, B., Xiong, D., and Su, J. Accelerating neural transformer via an average attention network. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, 2018.
- Zhang, L., Halber, M., and Rusinkiewicz, S. Accelerating large-kernel convolution using summed-area tables, 2019. URL <https://arxiv.org/abs/1906.11367>.